Particle Filtering for Location Estimation

Oliver F. Krenek B.E. (Hons)

Department of Electrical and Computer Engineering

A thesis submitted in partial fulfilment of the requirements for the degree of
Masters of Engineering in Electrical and Computer Engineering

University of Canterbury
Christchurch, New Zealand
April 2011

Supervisors: R. Y. Webb, P. J. Bones

Abstract

Vehicle location and tracking has a variety of commercial applications and none of the techniques currently used can provide accurate results in all situations. This thesis details a preliminary investigation into a new location estimation method which uses optical environmental data, gathered by the vehicle during motion, to locate and track vehicle positions by comparing said data to pre-recorded optical maps of the intended location space.

The design and implementation of an optical data recorder is presented. The map creation process is detailed and the location algorithm, based on a particle filter, is described in full. System tests were performed offline on a desktop PC using real world data collected by the data recorder and their results are presented. These tests show good performance for the system tracking the vehicle once its approximate location is determined. However locating a vehicle from scratch appears to be infeasible in a realistically large location space.

Acknowledgements

Firstly, I would like to thank both my supervisors Dr. Russell Y. Webb and Prof. Philip J. Bones. Their creative ideas, strong engineering skills, constant encouragement and endless patience were vital to the production of this thesis. I am enormously grateful to them both.

Secondly, I would like to thank the whole Department of Electrical and Computer Engineering for providing the excellent facilities and friendly work environment that I was privileged to be a part of throughout both my Masters and undergraduate degrees.

Finally, I would like to thank my friends and family, for their continued support and for helping make my student life an enjoyable experience.

Contents

Chapter 1

Introduction

This thesis presents a novel technique for locating and tracking vehicles called Navigation Using Low-resolution Light Sensing (NULLS). The key concept underpinning the project is that the object of interest collects optical data from its surroundings during motion and attempts to match this data with a location on a previously recorded ”optical map” of its intended environment. The objectives of the project were to develop a hardware platform for collecting optical data, to develop an algorithm for performing the location estimation and to test the viability of the concept in the context of locating and tracking a vehicle in an urban environment.

1.1 Motivation

Systems capable of locating and tracking vehicles in real-time have generated significant interest over the past two decades. The following is a list of most common applications of such systems.

- Location data - Location information and directions to desired locations can be given to the driver.

- Fleet management - For companies that manage fleets of vehicles, knowing the real-time location of all drivers allows management to meet customer needs more efficiently.

- Asset tracking - Companies needing to track valuable assets for insurance or other monitoring purposes can plot the real-time asset location on a map and closely monitor movement and operating status.

- Covert surveillance - Vehicle location devices attached covertly by law enforcement organisations can be used to track journeys made by individuals who are under surveillance.

- Stolen vehicle recovery - Vehicles can be outfitted with tracking units to allow police to do tracking and recovery.

1.2 System overview

NULLS equipped vehicles observe their environment by recording the output of a highly directional light sensor oriented perpendicularly to the vehicle's forward motion. The light sensor thus points to the left of the vehicle and measures the luminance of the scene on the side of the street within its very narrow field-of-view. Figure 1.1 show the positions of the light sensor and the measurement point. Luminance values are recorded at uniform time intervals. This means the vehicle's speed must also be recorded and fused with the luminance time series data to generate a spatial luminance signal. An example of such a signal is shown in Figure 1.2. This signal is used both to generate the ”luminance map” and as the ”navigation signal” which is compared to parts of the luminance map during the location estimation process.

The luminance of objects typically found on the side of a street (fences, hedges, houses etc) changes significantly due to factors such as time of day and cloud positions, and they are often obstructed from the light sensor's view by parked cars. This means the observation signals contain a large amount of variation from observation to observation. This variation

Figure 1.1 Photo of a car on a city street with the positions of the light sensor and measurement point marked

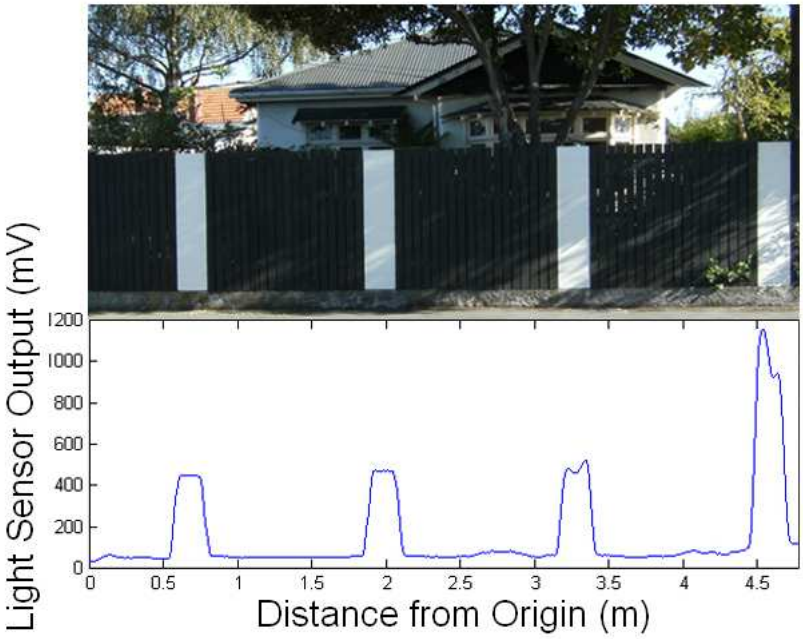


Figure 1.2 Example a spatial luminance signal

is loosely described as noise. Particle filtering is used to estimate the vehicle's location from the observation data as it has been shown to be particularly effective at coping with non-linear and non-Gaussian problems like this one [GGB+02].

1.2.1 A simple Example

Figure 1.3 illustrates how a very simple version of the NULLS would work. The map is limited to a single city block which is divided into discrete "blocks" of luminance.

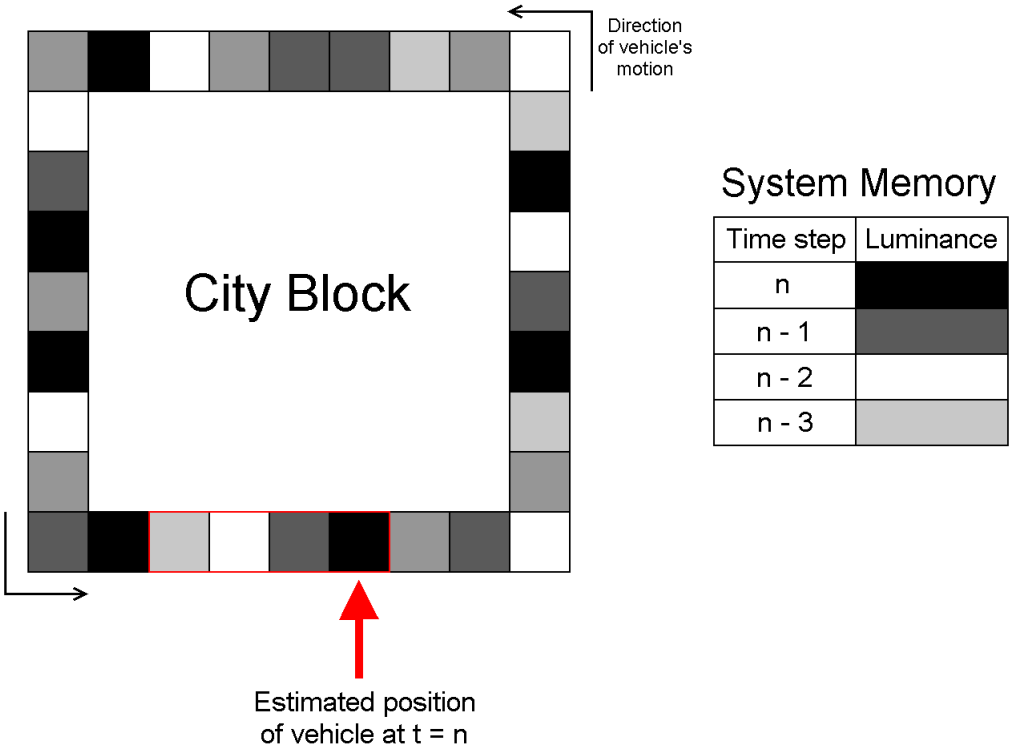


Figure 1.3 Simplified version of the NULLS problem

The problem in Figure 1.3 is clearly very easily solved. The following is a list describing the main reasons why the real NULLS problem is much more difficult.

- Large map - with a large quantity of map data, location cannot be achieved simply by checking every point on the map
- Sensor Noise - noise in the light and speed sensors leads to discrepancies between the map and measurement data
- Environmental Variations - Variations caused by traffic, weather and time of day lead to potentially very large discrepancies between the map and measurement data
- Alignment - sensor measurements cannot be easily alligned to the map data

1.2.2 Dataflow

NULLS operates in one of two distinct phases: the map recording phase and the location phase. During the map recording phase (see chapter 2 and 3), luminance and speed signals are combined to form a spatial luminance signal which forms part of the luminance map. During the location phase (see chapter 5), a spatial luminance signal is again generated from the luminance and speed time series data and is fed into a particle filter. The particle filter uses the observation signal and the previously recorded luminance map to estimate the vehicle's location's probability density function (PDF). The location estimator module outputs a location estimate, based on the current PDF provided by the particle filter. Figure 1.4 shows the flow of data through the NULLS system.

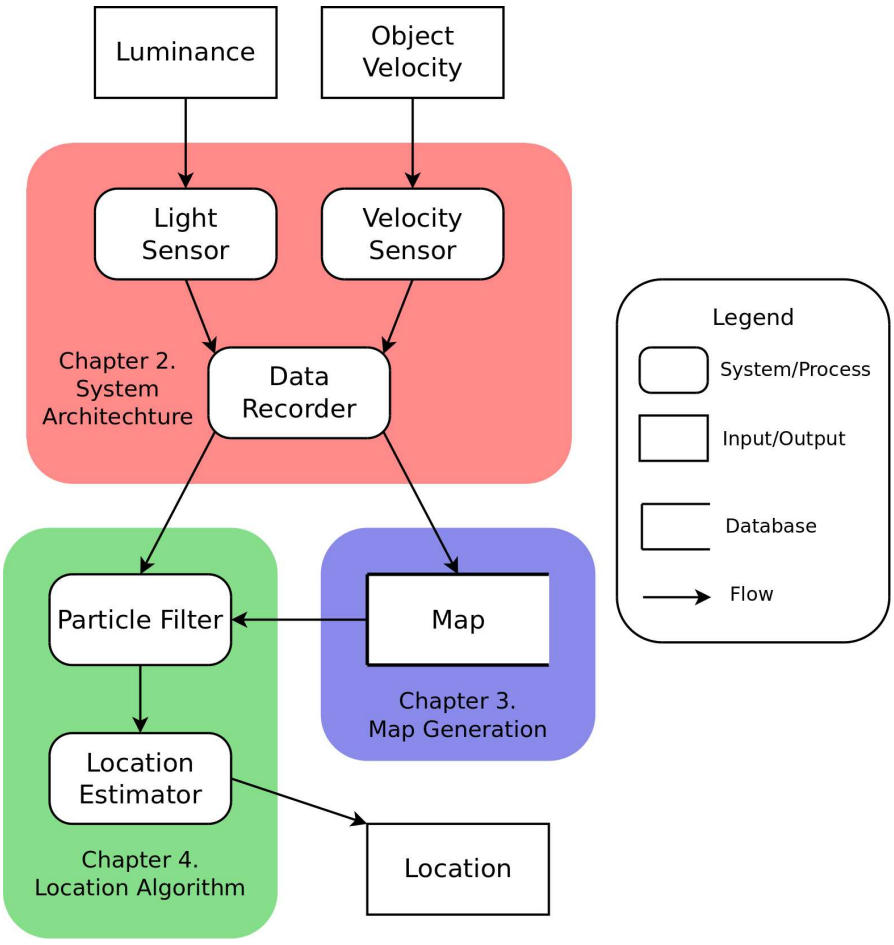


Figure 1.4 Data Flow Diagram of NULLS

1.3 Comparison with Other Navigation Systems

This section summaries GPS and Inertial Navigation Systems, the two most common vehicle tracking systems in use today, and compares their main characteristics with those of NULLS.

1.3.1 Global Positioning System

Global Positioning System (GPS) based navigation is certainly the most widespread navigation technique used today. GPS receivers are able to accurately determine their position by reading signals transmitted by GPS satellites. Each satellite continuously transmits messages containing the precise time the message was sent and the satellite's current position. GPS receivers require signals from four satellites because there are four unknowns in the position solution. The receiver's position accounts for three unknowns (x, y, and z coordinates). The last unknown is the time taken for the GPS signal to reach the receiver. This time is unknown because the receiver's clock is not synchronized with the GPS clocks. GPS based navigation systems suffer from one main disadvantage, known as the Urban Canyon phenomenon. In city environments, tall buildings can often block many if not all GPS satellite signals, preventing GPS receivers from computing their location, at least with their usual accuracy. For further details, see [GWA07], [Dan99], [Kel94] and [BF99].

1.3.2 Inertial Navigation

Inertial navigation systems (INS) use accelerometers and gyroscopes to track the position and orientation of an object relative to a known starting point, velocity, and orientation. Typically three orthogonal accelerometers and three orthogonal gyroscopes are used to measure the object's linear acceleration and angular velocity respectively. Orientation relative to the initial orientation is computed by integrating the angular velocity signal. Similarly, linear position relative to the starting position is computed by double integration of the linear acceleration signal. Since no accelerometer or gyroscope is perfectly accurate, all INS suffer from integration drift. Small inaccuracies in the acceleration and angular velocity signals lead to progressively larger errors in the computed position and orientation of

the object. An important advantage of INS is that once the initial position, velocity, and orientation of the object is known, no external reference is required, making them immune to jamming and deception. The system presented in this paper is complementary to INS, because it provides an absolute location estimation from which relative estimates can be accumulated. For further details see [GWA07], [BF99] and [Kel94].

1.3.3 Comparison

Table 1.1 compares some of the main characteristics of GPS-based navigation, INS and NULLS.

System	Accuracy	Speed	Works anywhere?	Self-contained?	Start location required?
GPS	High	Fast	No - Not in Urban Canyons	No - Requires GPS satellites	No
INS	Low - Errors Accumulate	Fast	Yes	Yes	Yes
NULLS	High	Slow to locate fast to track	No - Only in pre-recorded areas	Yes	No

Table 1.1 Comparison of Navigation Systems

1.4 Thesis Structure

The thesis is laid out as follows.

- **Chapter 2** details the development of the optical data collector including component choices, circuit schematics and software flowcharts.
- **Chapter 3** explains how the optical maps are generated from the raw optical and

speed data sets.

- **Chapter 4** summarises the general principles of particle filtering
- **Chapter 5** describes the NULLS location estimation algorithm
- **Chapter 6** presents the results of a number of tests on the viability of NULLS for both locating and tracking vehicles in urban environments.
- **Chapter 7** summarises the main findings of the project, and suggests a number of directions for further development of the system.

Chapter 2

System Architecture

2.1 Overview

This chapter describes the architecture of NULLS. The system is made up of three major components: the speed sensor, for measuring the vehicle's speed; the light sensor, for measuring luminance; the data processor, for recording the data provided by the two sensors. Luminance samples are taken at equally spaced points in time. This means the speed data is required to turn the luminance time-series signal into a spatial luminance signal. Figure 2.1 shows a top level block digram of the data collector.

Figure 2.1 Block Diagram of Data Collector

2.2 Light Sensor

The light sensor measures the luminance of the scene to the left of the vehicle and perpendicular to the vehicle's longitudinal axis. The sensor has a very narrow field of view (FOV). In the NULLS prototype, the FOV was initially limited by gathering light through a narrow, opaque tube. The FOV could thereby be adjusted by varying the length or radius of the tube. Such a small FOV meant that the recorded signal was an accurate representation of the actual luminance signal but it also required a very sensitive light sensor to make an accurate measurement. Later in development, the tube was replaced with a box containing a vertical slit through which light was gathered. The slit's lengthm and the distance between it and the light sensor was such that the scene visible to the light sensor was everything between the footpath and the top of the fence line. The recorded values were thus the average luminance of all the objects in this scene.

One advantage of using a slit is that the recorded luminance is less affected by the position of the vehicle on the road. The curvature of a typical city road across its width means that the angle which the optical axis makes with the horizontal changes as the distance of the vehicle from the centre line varies. The potential variation in this angle means a potentially different scene could be visible to the sensor during different recording sessions with the vehicle at the same location along its route. For the tube sensor this could mean a completely different object was measured, but for the slit sensor the two scenes would be largely the same.

The slits dimensions were 1mm x 80mm and the light sensor was positioned 20cm behind the slit point directly out. These dimensions lead to a detection area of 2m high and 2.5cm wide at a distance of 5m from the sensor (the approximate distance between a car on the road and the fence to its left).

The TSL257 [Sol07] was chosen for the NULLS prototype's light sensing component. It is a high sensitivity, low noise, light-to-voltage converter which combines a photodiode and a transimpedance amplifier on a single CMOS integrated circuit. Figure 2.2 shows a

function block diagram of the TSL257 and Figure 2.3 shows the very simple light sensor circuit used by the NULLS prototype. The following is a list of the main characteristics of the TSL257.

- High sensitivity
- Low noise - 200μV_{rms} DC to 1kHz
- High power supply ripple rejection - 35dB at 1kHz
- High linearity
- Focussing - the sensor is housed in transparent plastic with a convex spherical front surface to focus incident light onto the photodiode

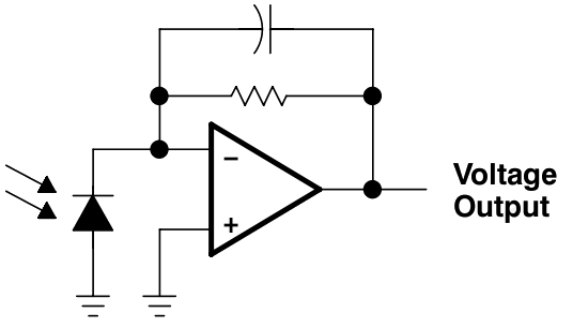


Figure 2.2 Functional block diagram of TSL257 [Sol07]

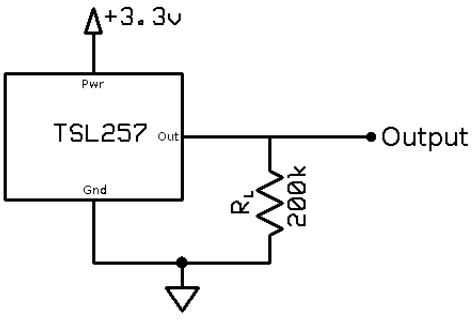


Figure 2.3 Light Sensor Circuit Schematic

2.3 Speed Sensor

2.3.1 Overview

The speed sensor measures the period of every rotation of one wheel on the vehicle. This data, along with the circumference of the vehicle's tyres, can be used to infer the distance travelled at any point in time during a recording session. The distance travelled time-series data is required to create a spatial luminance signal from the luminance samples which are recorded at regular intervals in time. The dataflow of the speed sensor is illustrated in Figure 2.5 In the prototype NULLS, wheel rotation was detected by suspending a magnetic field sensor above the wheel and attaching a permanent magnet to the wheel so that it passed past the sensor once per rotation. This setup is shown in Figure 2.4. The intervals between adjacent leading edges of the pulse from the magnetic field sensor were measured using a 1.5MHz timer. The tyre circumference was estimated by measuring the tyre diameter. If different vehicles were used for the map generation (see Chapter 3) and the location tests (see chapter 6 it would be important for the tyre diameter measurements to be highly accurate as errors would lead to distance data collected by one vehicle being linearly scaled compared with distance data collected by another vehicle. Since only one vehicle was used to gather data during this project approximate tyre diameter measurements were acceptable.

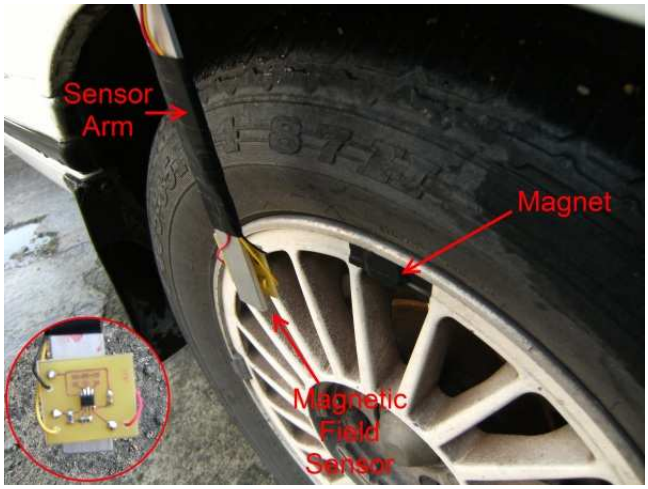


Figure 2.4 Photo of the magnet and the magnetic field sensor in relation to the vehicle's wheel

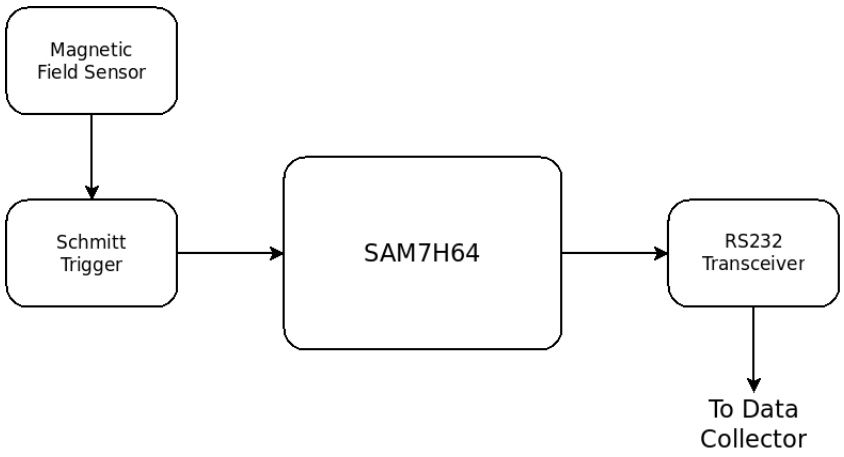


Figure 2.5 Speed sensor dataflow diagram

2.3.2 Hardware

This section details the hardware modules of the NULLS prototype speed sensor. The SAM7H64 is the speed sensor's microcontroller board and the magnetic field sensor and Schmitt trigger are used to detect wheel rotations.

SAM7H64

The speed sensor required a separate microcontroller board. This was because it required an interrupt to be triggered when the magnetic field sensor detected the magnet and the QWERK, a single board computer used to record the speed and light data (see section 2.4 for details), does not support interrupts as it runs Linux, a non-real-time operating system. The SAM7H64 [Oli06] was chosen to support the speed sensor as it featured a UART, for communicating with the QWERK, a GPIO interrupt support, for detecting wheel rotations, 16bit timers for measuring the period of the wheel rotations and because it could be powered by the QWERK through a USB cable.

Magnetic Field Sensor

The AD22151 [Dev03] was chosen for the magnetic field sensor. It is sensitive enough to produce a significant response when a small, neodymium magnet passes 5cm in front of it

and is therefore perfectly acceptable for this application. Figure 2.6 shows the schematic of the sensor circuit and Figure 2.7 shows an approximation of the sensor's response to the magnet passing 5cm in front of it.

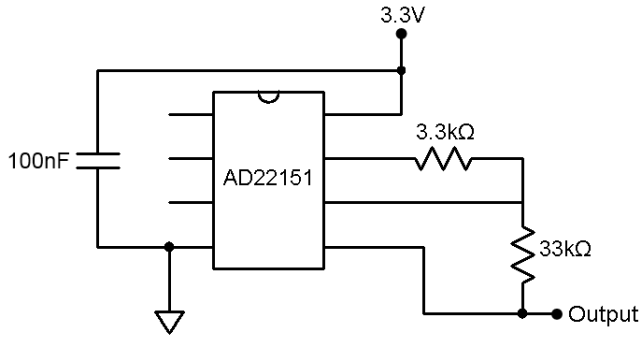


Figure 2.6 Magnetic field sensor circuit schematic

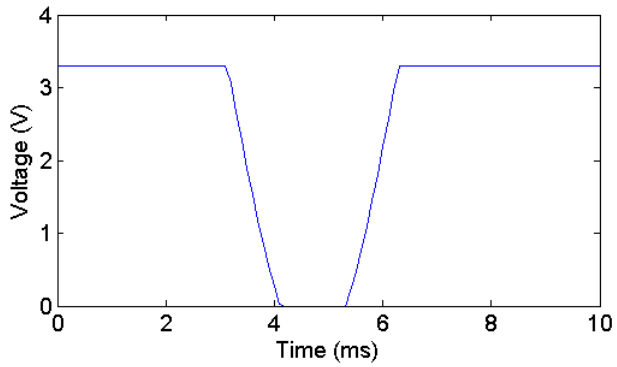


Figure 2.7 Simulated magnetic field sensor response to a magnet pass

Schmitt Trigger

The magnetic field sensor's output was passed through an inverting Schmitt trigger before arriving at the SAM7H64 to ensure that noise in the signal would not generate extra interrupts. The upper and lower thresholds were designed to be 1V apart. The circuit schematic is shown in Figure 2.8 and the approximate response to the magnetic sensor output signal in Figure 2.7 is given in Figure 2.9. Figures 2.7 and 2.9 were created in Matlab using data from the components' actual responses, which were measured using an oscilloscope.

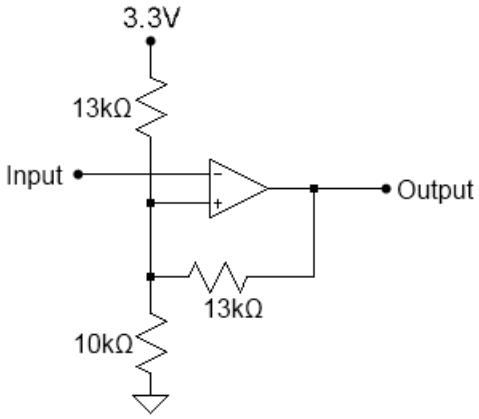


Figure 2.8 Magnetic field sensor circuit schematic

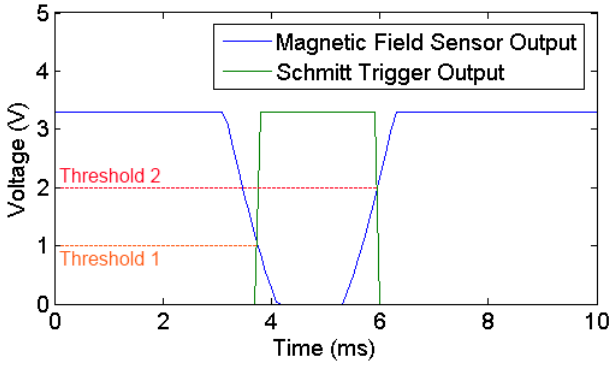


Figure 2.9 Simulated magnetic field sensor and Schmitt trigger response to a magnet pass

RS-232 Transceiver

The SAM7H64 required a separate RS-232 transceiver to convert its 0V to 3.3V UART signal into an RS-232 signal which could be received by the Data Processor. The MAX3232 [MAX99] was used for this task. The circuit required for its operations is shown in Figure 2.10 and is taken directly from the component's datasheet [MAX99]. T1IN and R1OUT were connected to the SAM7H64's UART transmit and UART receive ports respectively. Similarly T1OUT and R1IN were connected to the receive and transmit lines of the data processor's RS-232 port.

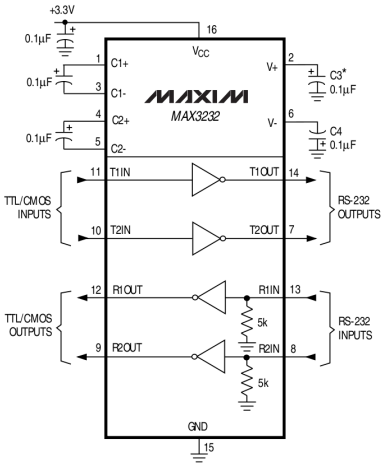


Figure 2.10 MAX3232 operating circuit [MAX99]

2.3.3 Software

The speed sensor software operates in two phases: the data collection phase, and the data transfer phase. During the data collection phase, the speed sensor measures the period of each wheel rotation and stores a sequence of time interval in its own memory. During the data transfer phase, the intervals are transmitted to the data processor. Figure 2.11 shows an overview of the speed sensor software but, because the software is interrupt driven, it is not a true representation of how the program works.

Interrupt Service Routines

Interrupt service routines are required to handle timer overflows, the ”end of interval” events and the UART communication with the data processor.

Timer Overflow ISR The timer overflow interrupt is generated when the 16bit timer reaches its maximum value of 0xFFFF. The timer overflow interrupt service routine simply adds this value to the 32bit variable which represents the interval currently being measured (called *CurrentInterval* in procedure 2.1) and resets the timer. *CurrentInterval* has a maximum value of 0xFFFFFFFF and the timer frequency was set to 1.5MHz. This meant a maximum interval of 2863s and a time resolution of 666.7ns.

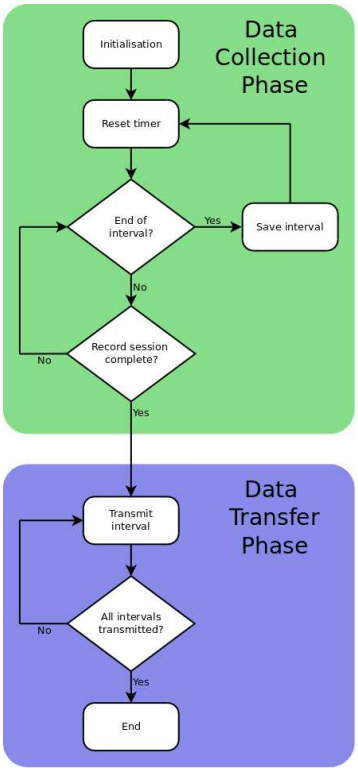


Figure 2.11 Flow chart of Speed Sensor Software

Procedure 2.1 Timer Overflow ISR

Clear Timer Overflow Interrupt
$CurrentInterval$ += 0xFFFF
Reset Timer

End of Interval ISR When the magnetic field sensor detects the permanent magnet pass beneath it, an interrupt is generated to signal that the current interval is now over. The end of interval ISR adds the current timer value to the interval currently being measured and stores the final interval value in memory. The timer is reset so that a new interval can be measured. See procedure 2.2.

UART ISRs The UART module on the SAM7H64 was used to communicate with the data processor. It generated two interrupts: the receive interrupt, generated when the module received a character from the data processor, and the transmit interrupt, generated whenever the module's transmit register was empty. The system worked in a half-duplex mode

Procedure 2.2 End of Interval ISR

```
Clear FIQ Interrupt
CurrentInterval += GetTimerValue()
Intervals[NumIntervals] = CurrentInterval
NumIntervals++
CurrentInterval = 0
Reset Timer
```

where only one of these interrupts was enabled at any one time. During the recording phase, the system stayed exclusively in the receive state, so that the data processor could signal the start and the end of a recording session by sending the characters 's' and 'e'. For the data transfer phase, the system alternated between the receive and the transmit states. The data processor would request an interval by sending an 'i' character to the SAM7H64 while it was the receive state. The SAM7H64 would then change to the transmit state and send the next interval in its memory. This process was continued until all the intervals had been sent. Procedures 2.3 and 2.4 describe the interrupt service routines for the two UART interrupts.

Procedure 2.3 UART Receive ISR

```
char = GetReceivedChar()
if char == 's' then
    Enable Timer Overflow Interrupt
    Enable End of Interval Interrupt
    numIntervals = 0
    CurrentInterval = 0
    Reset Timer
else if char == 'e' then
    Disable Timer Overflow Interrupt
    Disable End of Interval Interrupt
    Enable UART Transmit Interrupt
    Disable UART Receive Interrupt
    sendString = atoi(NumIntervals)
    charsSent = 0
else if char == 'i' then
    sendString = atoi(GetNextInterval())
    Enable UART Transmit Interrupt
    Disable UART Receive Interrupt
end if
```

Procedure 2.4 UART Transmit ISR

if $charsSent$ == $sendString$.length **then**
  Disable UART Receive Interrupt
  Enable UART Receive Interrupt
else
  $uartSendRegister$ = $sendString$[$charsSent$]
  $charsSent$++
end if

2.4 Data Processor

The data processor measures the voltage level of the light sensor's output periodically throughout the recording session. Upon completing a recording session, the data processor gathers the speed data from the speed sensor and stores both the light and speed datasets in its permanent memory. The operation of the data processor is shown in Figure 2.12.

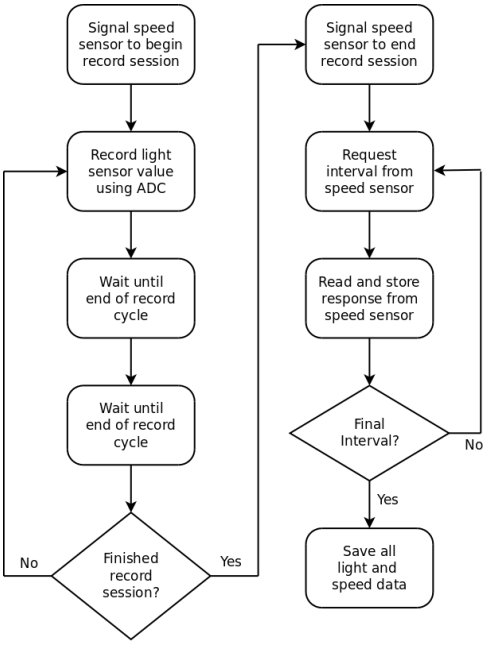


Figure 2.12 Data Processor Flow Chart

In the NULLS prototype, Charm Lab's QWERK [Lab06] was used for the data processor. The QWERK is a 200MHz single-board-computer that runs Linux. The light sensor voltage was measured using the QWERK's 12-bit ADCs with a cycle period of 6ms. This period resulted in a distance of 8.3cm between adjacent luminance values when the vehicle was

travelling at 50km/hr. A short cycle period would have provided more freedom to the data analysis, but 6ms was nearing the limit of what the QWERK could reliably achieve. The speed data was received through one of the QWERK's two RS-232 ports. The QWERK was configured to run the data collecting software on startup, and the start and end of a recording session was signalled using two push buttons connected to two of the QWERK's digital inputs.

Chapter 3

Map Generation

This chapter describes the process of transforming the speed and luminance data into a spatial luminance map on which the vehicle can be located.

The raw data is a list of wheel rotation intervals and a list of luminance values sampled at a fixed rate. The output data is a list of luminance values spaced equally in distance. Figure 3.1 illustrates the processing of the two input signals into a single output sequence.

The process illustrated in Figure 3.1 creates a luminance map for one side of a section of a city street. Sections 3.1, 3.2 and 3.3 describe the steps in the process. A two dimensional map is created by linking several of these single sided street sections together in a graph structure. The linking process is explained in section 3.4.

3.1 Processing Speed Data

The raw speed data is a series of wheel-rotation intervals. Each interval is the time it takes for the vehicle to move the length of the wheel's circumference (~2m). Figure 3.2 shows a small example dataset for this point in the process.

The first step in processing the data is to accumulate the signal so that each value in the

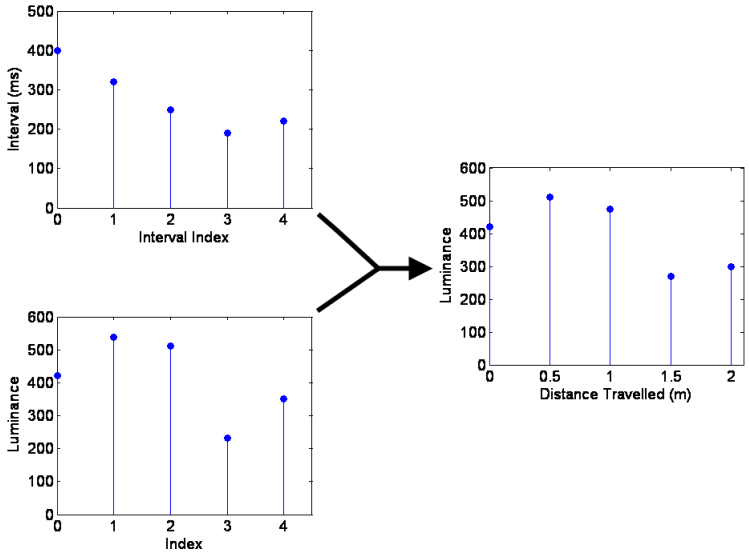


Figure 3.1 The luminance map for one side of a city street is formed by combining two sets of input data into a single sequence of luminance sampled at equal distance intervals

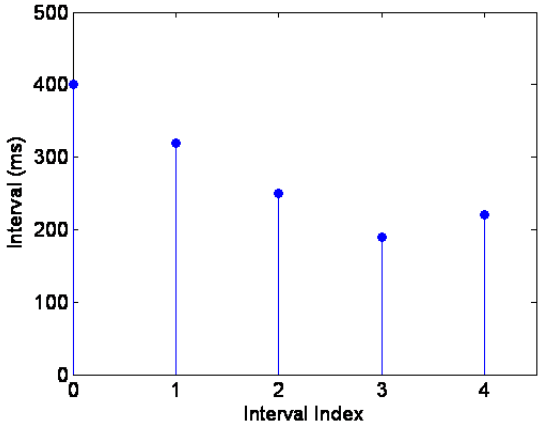


Figure 3.2 Example of raw interval data

processed dataset is the sum of the equivalent raw data value and all previous raw data values. The new dataset represents the time taken for the vehicle to make a certain number of wheel rotations. Figure 3.3 shows the data from Figure 3.2 after the accumulation process.

To convert this ”wheel rotations” time series data into a distance travelled time series (speed data), we simply multiply the wheel rotation index by the circumference of the

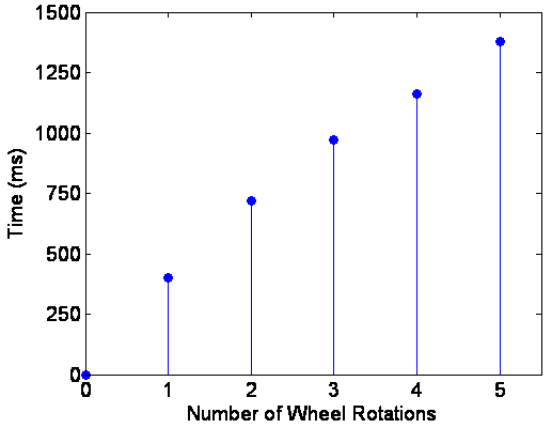


Figure 3.3 Example of accumulated interval data

vehicle's tyre. Figure 3.4 shows the example data after this process. Note that the x and y axes have been reversed to present the distance travelled time-series data in the typical manner.

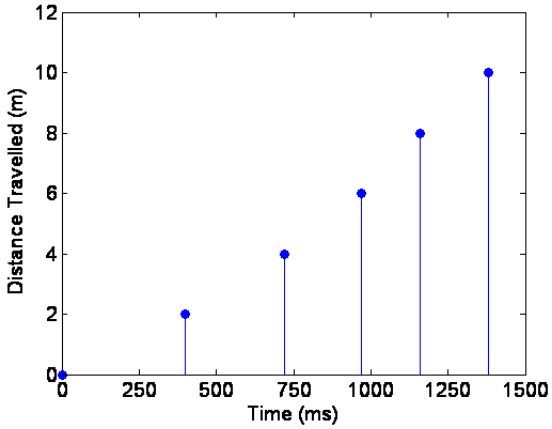


Figure 3.4 Example of speed data

3.2 Positioning Luminance Data

This section involves finding a "distance from origin" value for each luminance value in the data set. The sampling rate of the luminance data is constant so the time since the start of the recording session of each luminance value is the value's index multiplied by the sampling period. Figure 3.5 illustrates this process using a sampling period of 300ms.

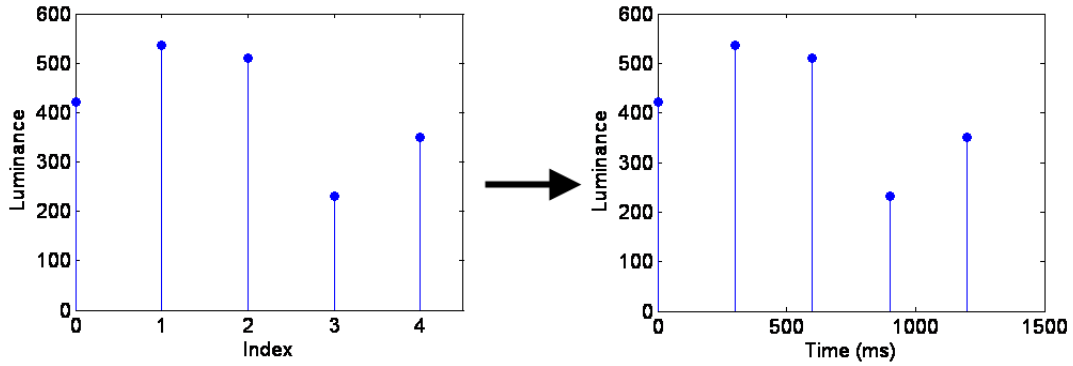


Figure 3.5 Example of luminance time-series data

At this point, the time when each luminance sample was recorded is known. The distance travelled time series data can therefore be used to determine the distance travelled value of each luminance sample. Because the distance travelled is only known at the start of each wheel rotation, linear interpolation is used to estimate the position of each luminance sample within the ∼2m interval it falls within. This process is illustrated in Figure 3.6. The signals produced by this process contain luminance data values at known positions and are referred to as luminance spatial signals.

3.3 Processing Luminance Signal

In order for the map creator to control the memory requirements of the maps, the luminance spatial signals require some processing.

Firstly, a constant data spacing is achieved by linear interpolation of the luminance spatial signal at the desired positions. For example, if the desired data spacing is 1m, then luminance values will be interpolated from the original signal at positions: 0m, 1m, 2m, etc. This saves memory as not only is the sampling rate now under control of the map creator, but the position values of the luminance data are no longer required because they can be determined from the spacing value and their indices within their data structure. Secondly, the luminance values are further quantized to reduce the amount of memory required.

An example of this process is illustrated in Figure 3.7 with a data spacing of 0.5m and

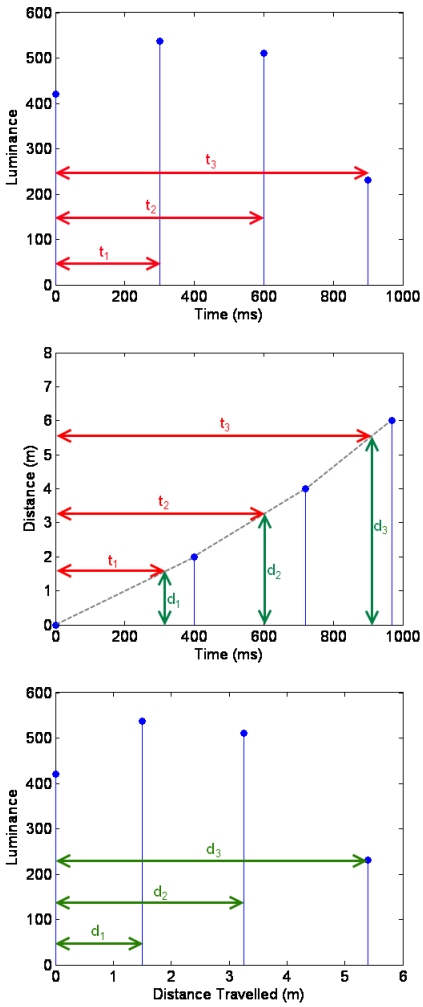


Figure 3.6 Example of the luminance data positioning process - the position in time of each luminance sample is known and the distance travelled time-series data is used to find the distance travelled value for each sample by interpolation.

quantization level separation of 100. In section 6.3.2 the results of an investigation into how varying the data spacing and degree of quantization affects system performance are presented.

3.4 Two Dimensional Mapping

The spatial luminance signals generated by the processes described in the previous sections are used to map one-sided sections of streets and will be referred to as *street segments*.

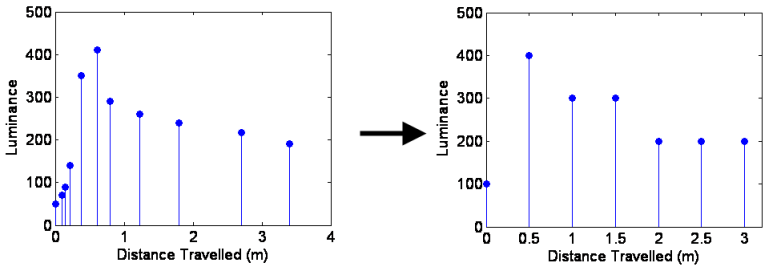


Figure 3.7 Example of the luminance data processing - the data spacing 0.5m and the quantization levels are 100 units apart

Because they only represent one side of a road, a vehicle travelling along a street segment will always be moving away from the origin and towards the end point. Multiple street segments can be linked together to form 2-dimensional maps of any complexity. Each street segment generally maps a section of street between intersections. This means that once on a street segment, a vehicle has no choice but to follow it to its end (U-turns are not considered) where it will find an intersection. Figure 3.8 shows a very simple luminance map with only three street segments and one intersection with more than one path available.

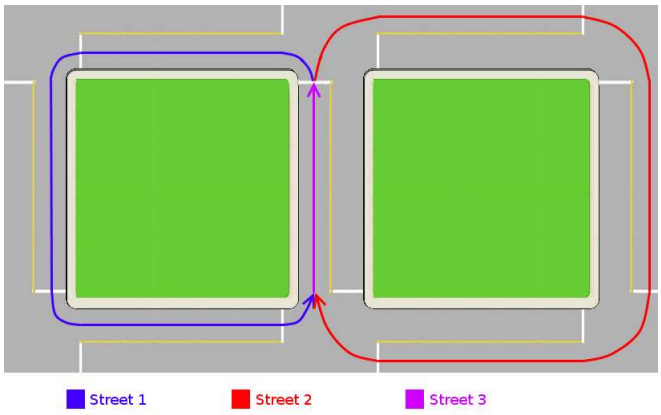


Figure 3.8 A simple example of a 2D luminance map

Street segments contain the following data:

- **ID** A unique identifier
- **Data spacing value** The distance between two adjacent luminance values

- **Luminance values** A list containing all of the luminance values
- **Connect to** A list of street segment IDs defining which street segments are connected to the end of this street segment
- **Connect from** A list of street segment IDs defining which street segments are connected to the start of this street segment

The data for the street segments in Figure 3.8 is given in Table 3.1. This data is provided primarily to clarify how the ”Connect to” and ”Connect from” lists determine how the street segments in a 2-dimensional luminance map are linked together.

ID	Data spacing value	Luminance values	Connect to	Connect from
Street1	0.5m	120,321,422...	Street3	Street3
Street2	0.5m	242,122,120...	Street3	Street3
Street3	0.5m	231,221,340...	Street1, Street2	Street1, Street2

Table 3.1 Example street segment data

Chapter 4

Particle Filters

This chapter provide a brief overview of particle filters. For a detailed explanation see [DKZ+03], [DdFG01], [GGB+02] and [KFM04].

4.1 Overview

Particle filters estimate the state of dynamic systems from sensor data. They use a sequential Monte Carlo methodology where relevant probability distributions are recursively computed using the concept of approximation of probability distributions using discrete random measures [DKZ+03]. Over the last two decades particles filters have been applied with great success to a variety of state estimation problems including visual tracking, speech recognition, and mobile robotics [DdFG01]. The more non-linear the model, or the more non-Gaussian the noise, the more potential particle filters have, especially in applications where computational power is relatively cheap and the sampling rate moderate [GGB+02].

4.2 Models

A large portion of the theory on sequential signal processing is about signals and systems that are represent by state-space and observation equations [DKZ+03]. These equations take the form shown below.

$$x_t = f_t(x_{t-1}, u_t) \tag{4.1}$$

$$y_t = g_t(x_t, v_t) \tag{4.2}$$

where y_t is a vector of observations

x_t is a state vector

$g_t(.)$ is a measurement function

$f_i(.)$ is a system transition function

u_t and v_t are noise vectors

t is the time step index

Equations 4.1 and 4.2 are known as state and observation equations respectively. The standard assumptions are that x_0, u_t and v_t have independent and known probability densities of p_{x_0}, p_{u_t} and p_{v_t} respectively.

Particle filters approximate distributions using discrete random measures defined by particles and weights assigned to particles. If $p(x)$ is the distribution of interest then its approximating random measure is given by the particles $x^{(m)}$ with weights $w^{(m)}$ where m ranges from 1 to M, the number of particles in the approximation. Distribution $p(x)$ is approximated with equation 4.3.

$$p(x) = \sum_{m=1}^{M} w^{(m)} \delta(x - x^{(m)}) \tag{4.3}$$

where $\delta(.)$ is the Dirac delta function

4.3 Algorithm

The particle filtering algorithm can be broken down into the following steps [GGB+02]:

1. Initialisation
2. Measurement Update
3. Resampling
4. Prediction
5. Iterate to step 2

1. Initialisation Generate M particles x_0^m from the distribution p_{x_0} and assign initial weights w_0^m of $\frac{1}{M}$

2. Measurement Update Read in a new data value y_t and update the particle weights using equation 4.4. In words the term $p(y_t|x_t^m)$ means the probability of measuring the value y_t given the system is in state x_t^m.

$$w_t^m = w_{t-1}^m p(y_t|x_t^m), m = 1, 2, ..., M \tag{4.4}$$

Normalise weights using equation 4.5 and approximate system state using equation 4.6.

$$w_t^m = \frac{w_t^m}{\sum_{i=1}^{M} w_t^i}, m = 1, 2, ..., M \tag{4.5}$$

$$\hat{x}_t \approx \sum_{m=1}^{M} w_t^m x_t^m \tag{4.6}$$

3. Resampling *a) Bayesian Bootstrap*

Take M samples with replacement from set $\{x_t^i\}_{i=1}^N$ where the probability to take sample i is w_t^i. This step is also called sampling importance resampling (SIR).

b) Importance Sampling

Only resample as above when the effective number of samples, given by equation 4.7 is less than a threshold M_{th}. Here, $1 \leq N_{eff} \leq M$, where the upper bound is attained when all particles have the same weight, and the lower bound when all probability mass is at one particle. M_{th} is often chosen to be $2M/3$.

$$M_{eff} = \frac{1}{\sum_{m=1}^{M}(w_t^m)^2} < M_{th} \tag{4.7}$$

4. Prediction Predict the new state vector x_{t+1} using the current state x_t, equations 4.1 and 4.2 and the known probability densities p_{x_0}, p_{u_t} and p_{v_t}.

5. Iteration Let $t = t + 1$, and iterate to step 2.

The main reason for resampling is to prevent high concentration of probability mass at a few particles. Without this step the weight of one of the particles will converge to 1 and the filter brakes down to a pure simulation [GGB$^+$02].

Chapter 5

Location Algorithm

This chapter details the algorithms used by the NULLS system.

5.1 Overview

NULLS uses a particle filter to perform the location estimation. The major steps involved are shown in Figure 5.1. Details of these steps are presented in sections 5.4 and 5.5. Section 5.4 describes an algorithm designed to solve a simplified version of the location problem where the location space is limited to a single loop in a city environment. This reduces the 2-dimensional location problem to one dimension where the location can be described as the distance from the origin. This is useful not only for helping the reader gain an understanding of the algorithm but also for presenting the results of the experiments found in chapter 6, as 1-dimensional location can be shown with significantly more clarity while retaining most of the problems associated with 2-dimensional location. Section 5.5 describes the full, 2-dimensional algorithm where the location space is not limited. The particles in the filter represent possible locations of the vehicle and their weights are estimates of the probability that the particle is at the vehicle's true location.

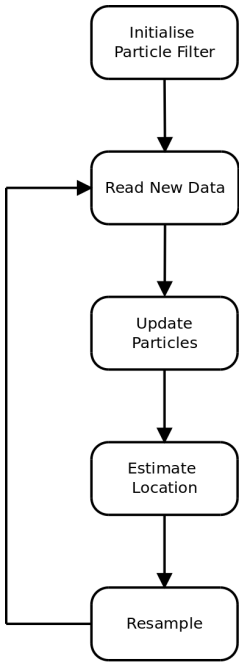


Figure 5.1 Flow chart of NULLS algorithm

5.2 Terms and system parameters used in this chapter

5.2.1 Terms

- **Navigation data** The luminance data collected by the vehicle as its location is being estimated by the system. Individual values are referred to as *test points*. Each test point has a distance value associated with it describing how far behind the vehicle the point was recorded at the current time step.

- **Map data** The luminance data previously collected to act as the map on which the vehicle's location is estimated.

- **Street** Refers to the location space used by the simplified or 1-dimensional algorithms.

- **Street segment** A 1-dimensional path that forms part of a 2-dimensional location space used by the full or 2D algorithm. See section 3.4 for details.

5.2.2 System Parameters

The following list describes the system parameters that are used often throughout this chapter. Less common ones are describes as they are used.

- **NUM_PARTICLES** The number of particles in the system.
- **MAX_NAVIGATION_DATA** The maximum number of test points stored by the system in the navigation data vector.
- **NAVIGATION_DATA_SPACING** The distance between adjacent test points.

5.3 Algorithm assumptions and key decisions

The key assumption made for this location problem was that the lower the difference between the navigation data, and the map data at a particular position, the more likely the vehicle is to be at that position and that no other conclusions can be drawn. This assumption is the basis for the particle update methods described in sections 5.4.3 and 5.5.3.

Both versions of the algorithm use the Bayesian Bootstrap method of resampling (see section 4.3) where resampling occurs on every system cycle. This decision was made because meaningfully high particle weights only occur very close to the vehicle so it is important to have a high rate of particle motion for it to be located in a timely fashion.

Unlike in most particle filters particle weights are never normalised. This is because the resampling routines (see sections 5.4.5 and 5.5.5) rely on knowing the magnitudes of the particle weights rather than just their relative weights. In short, normalising in this system would be discarding useful data.

5.4 Simplified Algorithm

This section presents a simplified version of the NULLS algorithm where the location space is limited to a single closed loop in an urban environment. Since the loop is one sided, a

location within the loop can be described by a single distance-from-origin value, making it a 1-dimensional problem. Figure 5.2 shows a one-block example of a location space suitable for the simplified algorithm.

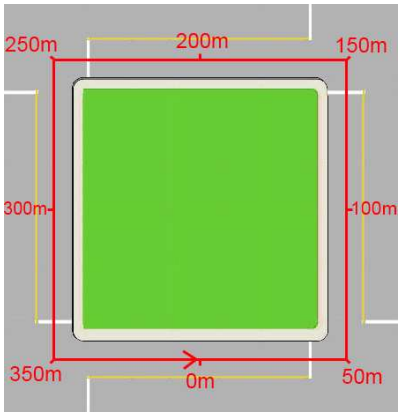


Figure 5.2 Example location space for the simplified algorithm

5.4.1 Initialisation

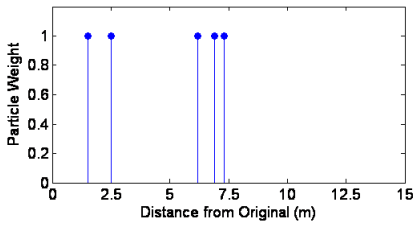
A set of particles is created and distributed randomly over the street. Particle positions are randomly calculated from a uniform distribution between 0 and the length of the street. The initial weight of the particles is arbitrary as an update is always performed before the particle weights are read. Updates involve new weight values for all particles which are not based on any previous weight values.

5.4.2 Data Measurement

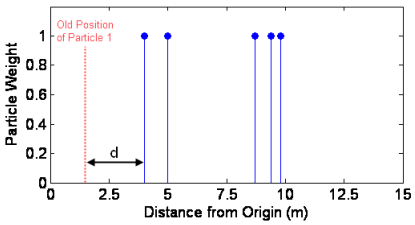
When the vehicle has travelled the distance, defined by the system parameter *NAVIGATION_DATA_SPACING*, since the previous data measurement a new test point is recorded from the light sensor. It is added to the front of the navigation data vector with a distance value of zero. The remaining test points have *NAVIGATION_DATA_SPACING* added to their distance values. If the size of the navigation data vector exceeds system parameter *MAX_NAVIGATION_DATA* the oldest element is discarded.

5.4.3 Update

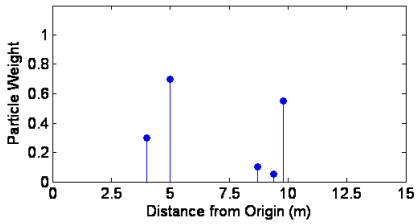
The position of each particle is shifted along the street in the positive direction by *NAVIGATION_DATA_SPACING*, the distance the vehicle has travelled since the last update. This means that each particle is in the same position relative to the vehicle as it was at the end of the previous cycle. The weight of each particle is then updated by comparing the navigation data to the map data at the particle's position. Map values at required positions are found by linear interpolation between the two data values closest to the desired position. Figure 5.3 shows a simple example of five particles being updated.



(a) Before update



(b) After particle shift (d = *NAVIGATION_DATA_SPACING*)



(c) After weight update

Figure 5.3 Example of the particle update process showing the positions and weights of 5 particles

Equation 5.1 shows the detail of how the particle weight is updated.

$$w_i = \frac{1}{1 + \left(\frac{\sum_{j=1}^{H} |n_j - m_{ji}|}{H} \right)} \tag{5.1}$$

where w_i is the new weight of the i^{th} particle

n_j is the j^{th} navigation data value recorded at distance d_j from the vehicle's current position

m_{ji} is the interpolated map value at distance d_j from the i^{th} particle's position

H is the number of navigation data values

Figure 5.4 clarifies the relationship between the terms: d_j, n_j and m_{ji} in equation 5.1 and shows how luminance values are linearly interpolated from the map data.

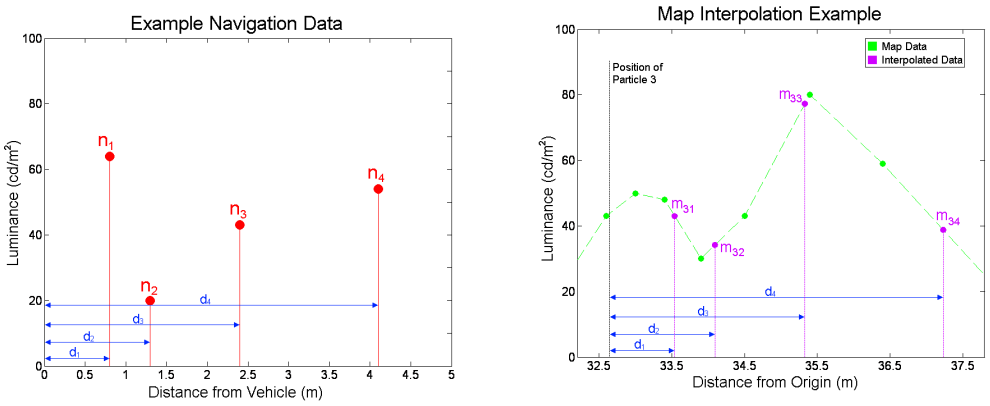


Figure 5.4 Visual representation of the terms used in equation 5.1

Algorithm 5.1 contains pseudo-code showing how a single particle is updated.

Algorithm 5.1 Particle Update

$totalDifference = 0$
$particle.position+=$ navigation data spacing
for i = 1 to $nav\_data.size$ **do**
  $position = particle.position - navData[i].distance$
  $totalDifference+= abs(navData[i].luminance - luminanceOnMap(position))$
end for
$averageDifference = totalDifference/nav\_data.size$
$particle.weight = 1/(1 + averageDifference)$

$luminanceOnMap(position)$ returns the luminance value interpolated from the map at *position*

5.4.4 Estimation

Two estimation methods were implemented to estimate the vehicle's location based on the current state of the particle filter. The results of a comparison between these methods are presented in section 6.3.1.

Maximum Weight Particle

The simplest method of estimating the vehicle's position is to find the particle with the highest weight and use its position as the estimate.

Weighted Average

The weighted average method of location estimation involves finding the average of all the particle positions where the importance placed on each particle's position value is directly proportional to the particle's weight. Higher weight particles are considered more important in this calculation than lower weight particles. In the case of the 1-dimensional closed loop location space, the weighted average is calculated by converting each particle into a vector, with it's angle representing its position on the street ($0\,^\circ$ refers to the street origin and $360\,^\circ$ to the street's endpoint) and its magnitude equal to its weight. The particle-vectors are then summed together and the resulting vector is converted back to a regular particle structure. The position of this new particle is the weighted average of all the other ones. This process is illustrated in algorithm 5.2.

Algorithm 5.2 Weighted Average

$vectorTotal.angle = 0$
$vectorTotal.length = 0$
for $i = 1$ to $NUM\_PARTICLES$ **do**
  $vectorCurrent.angle = (particleSet[i].position * 2 * PI)/$(length of street)
  $vectorCurrent.length = particleSet[i].weight$
  $vectorTotal += vectorCurrent$
end for
$estimatedPosition = (vectorTotal.angle *$ length of street$)/(2 * PI)$

5.4.5 Resampling

Resampling is the process of redistributing the particles over the map. The ultimate goal is to concentrate many particles around areas likely to contain the vehicle's true location and to shift others around the map so that a wide range of locations are tested. The first step is to randomly select *NUM_PARTICLES* particles with replacement according to their weights. "With replacement" means particles can be selected more than once and "according to their weights" means particles with higher weights are linearly more likely to be selected than particles with lower weights. Favouring high weight particles in this step leads to particles being concentrated around areas of interest. The current positions of these randomly selected particles form the basis for the new positions of the particles. New particle positions are calculated using equation 5.2.

$$p_i^* = N(p_i, \frac{\alpha}{w_i}) \tag{5.2}$$

where $N(\mu, \sigma^2)$ is a normal distribution with a mean of μ and a variance of σ^2

p_i^* is the i^{th} new particle position

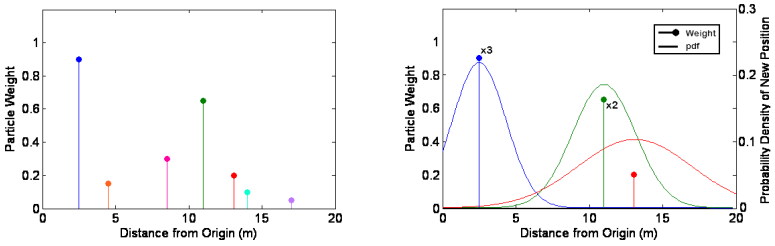
p_i is the position of the i^{th} randomly selected particle

w_i is the weight of the i^{th} randomly selected particle

α is a system parameter

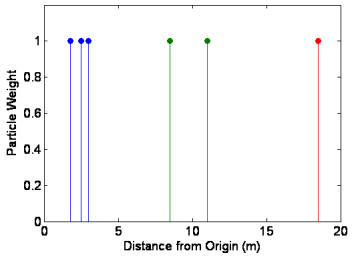
The system parameter alpha is used to control the scale of the particle movement. A higher value of alpha means a higher variance in the Normal distribution for a given particle weight and therefore larger average shift. Equation 5.2 shows that the variance of the particle shift is inversely proportional to the particle's weight. This means that high weight particles, which represent likely vehicle locations, are shifted by smaller distances on average compared to low weight particles.

Figure 5.5 illustrates the resampling process. Figure 5.5(a) shows six particles before the resampling process has begun. Figure 5.5(b) shows the particles after the selection process. The blue particle was selected three time, the green particle twice, and the red particle once. The other particles were discarded. The curves in this figure show the probability density functions of the selected particles' Normal distributions to be used when calculating the new positions. Figure 5.5(c) shows the new particles with their colours showing which of the original particles their position is based on.



(a) Before resample

(b) Particles are selected



(c) New particle positions are calculated

Figure 5.5 Example of the resampling process ($\alpha = 0.3$)

Algorithm 5.3 shows the resampling operation in the form of pseudo-code.

5.5 Full Algorithm

This section describes the full 2-dimensional algorithm. Generally only the advancements made on the simplified algorithm are presented. Particles used in the simplified algorithm

Algorithm 5.3 Resampling

$newPositions[NUM\_PARTICLES]$
for $i = 1$ to $NUM\_PARTICLES$ **do**
  $particle = getRandomParticle()$
  $variance = alpha/(particle.weight)$
  $mean = particle.position$
  $newPositions[i] = randomGaussian(mean, variance)$
end for
for $i = 1$ to $NUM\_PARTICLES$ **do**
  $particleSet[i].position = newPositions[i]$
end for

$getRandomParticle()$ returns a randomly selected particle where the probability of a particle being selected is directly proportional to its weight

contain only two data members: their weight and position on the street. In this section particles contain: their weight, the ID of the street segment they are on and their position on that street segment.

5.5.1 Initialisation

The random distribution of particles must now consider which street segment to place each particle on. This is randomly selected with the probability of a particle being placed on a particular street segment being proportional to that segment's length.

5.5.2 Measurement

This step is exactly the same as in the simplified algorithm as it only involves the distance the car has travelled between system cycles and the newly recorded luminance value.

5.5.3 Update

The update step is the same as in the simplified algorithm except for two cases. Firstly, when the particle being updated is near the start of the street segments it on. In this case "near" means within *NAVIGATION_DATA_SPACING*MAX_NAVIGATION_DATA*. When

this is the case the particle's luminance comparison must consider at least part of each of the street segments that connect to the start of the particle's street segment. Each possible path is tested and the one which leads to the lowest difference between map and navigation data is used to determine the particle's new weight. This process is illustrated in algorithm 5.4. Secondly, when a particle is shifted off the end of a street segment it randomly selects a connecting segment to be placed on. This leads to the desirable behaviour of concentrated particles splitting off onto different connecting streets (in approximately equal numbers) when the vehicle passes through an intersection.

Algorithm 5.4 Particle Update

$particle.position+=$ navigation data spacing
if $position$ is outside the boundaries of $particle.street$ **then**
  $particle.position, particle.street = wrapToRandomLinkedStreet(position, particle.street)$
end if
$paths[] = getPaths(particle.streetID)$
$totalDifferences[]$
for $i = 1$ to $paths.size$ **do**
  for $j = 1$ to $nav\_data.size$ **do**
    $position = particle.position - nav\_data[j].distance$
    $totalDifferences[i]+= abs(nav\_data[j].luminance - luminanceOnMap(position, paths[i]))$
  end for
end for
$minTotalDifference = findMin(totalDifferences)$
$averageDifference = minTotalDifference/nav\_data.size$
$particle.weight = 1/(1 + averageDifference)$

$luminanceOnMap(position, streetID)$ returns the luminance value interpolated from the map at *position* on segment *streetID*

$getPaths(streetID)$ returns a vector containing the street segments IDs of the segments which connect to the start of segment *streetID*

$wrapToRandomLinkedStreet(position, particle.street)$ returns a position and street segment ID for a particle that has been shifted off its original street segment (*particle.street*) to a randomly selected connecting street segment

5.5.4 Estimation

Max Weight Particle

This is the same as its equivalent in the simplified algorithm except that the estimate location is described by the maximum weight particle's street segment and position rather than just its position.

Weighted Average

This estimation method was not implemented in the full algorithm as it appeared to be highly complex with a large computational load and preliminary comparison tests between the two methods used by the simplified algorithm showed the maximum weight particle method to be generally superior.

5.5.5 Resampling

This step has the same basic concept as the simplified version except when a particle is shifted off the end of its street segment it randomly selects a connecting segment to be placed on.

Algorithm 5.5 Resampling

$newPositions[NUM\_PARTICLES]$
$newStreetIDs[NUM\_PARTICLES]$
for $i = 1$ to $NUM\_PARTICLES$ **do**
  $particle = getRandomParticle()$
  $variance = alpha/(particle.weight)$
  $mean = particle.position$
  $position = randomGaussian(mean, variance)$
  if $position$ is outside the boundaries of $particle.street$ **then**
    $newPositions[i], newStreetIDs[i] = wrapToRandomLinkedStreet(position, particle.street)$
  else
    $newPositions[i] = position$
    $newStreetIDs[i] = particle.street$
  end if
end for
for $i = 1$ to $NUM\_PARTICLES$ **do**
  $particleSet[i].position = newPositions[i]$
  $particleSet[i].street = newStreetIDs[i]$
end for

$wrapToRandomLinkedStreet(position, particle.street)$ returns a position and street segment ID for a particle that has been shifted off its original street segment (*particle.street*) to a randomly selected connecting street segment

$getRandomParticle()$ returns a randomly selected particle where the probability of a particle being selected is directly proportional to its weight

Chapter 6

Results and Discussion

This chapter presents the results of a series of tests run offline on a desktop computer using data gathered by the NULLS prototype. The experimental method and performance measures are defined and simple 1-dimensional location estimation and particle filter operation is demonstrated. The effect of varying system parameters on performance is investigated and the system is tested in different weather conditions and times of day. Finally 2-dimensional location estimation is demonstrated.

6.1 Experimental Method

The majority of the tests in this chapter use the NULLS system's simplified algorithm (see section 5.4 for details). This is because most of the problems associated with 2-dimensional NULLS location are present in the 1-dimensional version. The exception to this is branching, where the system must determine which route the vehicle has taken after arriving at an intersection. Branching is demonstrated in section 6.5.

Tests were performed on two sets of luminance data recorded from the same physical, urban area, one to act as the map and the other to act as the navigation data. The entirety of the map dataset was known to the system throughout the analysis while the navigation

data was read gradually as if it were being collected during the test. Before being used in systems tests the map dataset undergoes all the processing described in chapter 3 leaving it as a list (or series of lists in the 2-dimensional case) of quantized luminance values with an equal and known distance between them. The navigation dataset undergoes the processing in chapter 3 up to and including section 3.2 leaving it as a series of unquantized and unevenly spaced luminance values with known distances between them (note that quantization does occur on these luminance values when they are measured by the ADC). The reason for the differing treatment to the two datasets is to provide the offline testing software with data as close as possible to that which it would receive if it were performing actual real-time location. In the 1-dimensional tests, datasets were recorded from single city blocks while in the 2-dimensional test case the location space consists of one side of the streets around two blocks (see section 6.5 for details).

A NULLS test involved a simulation of the vehicle making one or more circuits of the location space. Each system cycle produced a location estimate which was compared to the vehicle's true location, and the distance between the two values was recorded. These recorded distances made up the test's *estimate error* signal. Analysis of the estimate error signal produced the performance measures described in section 6.1.2. When the NULLS system's performance is measured as it is in sections 6.3 and 6.4, rather than its operation being demonstrated as in sections 6.2 and 6.5, the tests are performed 100 times each and the results are averaged. Each trial starts the vehicle at a different random location. The system and testing parameters that are used throughout this chapter, unless otherwise stated, are provided in section 6.1.3.

6.1.1 Terms used in this chapter

- **Estimate error** - The estimate error is the distance between the estimated location and the vehicle's true location.

- **Location** - Location has occurred or the vehicle has been located when the estimate error has dropped below 2m and stayed below 2m for at least 10m.

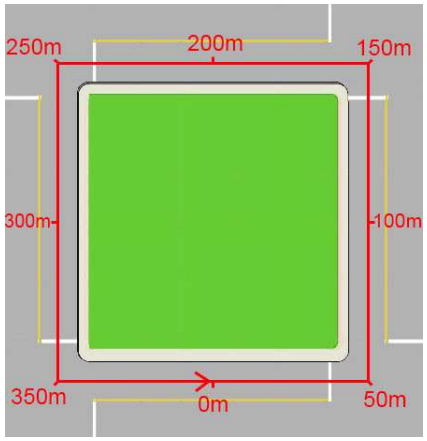


Figure 6.1 Illustration of the location space used in 1D tests

- **Tracking** - Tracking is occurring or the vehicle is being tracked if the estimate error is less than 2m and the location has occurred.
- **Deviation** - The system is deviating if location has previously occurred but the estimate error is now above the 2m threshold.
- **Navigation data** - The luminance data collected by the vehicle as its location is being estimated by the system.
- **Map data** The luminance data previously collected to act as the map on which the vehicle's location is estimated.
- **Test point** - A data point from the navigation dataset which is compared to the map during a particle update. Each test point has a distance value associated with it describing how far behind the vehicle the point was measure at the current time step.

6.1.2 Performance measures

The following list describes six quantitative measures for the performance of the NULLS system.

- **Average error** - The average estimate error throughout the entire test. A simple mea-

sure for the system's overall performance.

- **Tracking error** - The average estimate error while the system is tracking the vehicle
- **Location Distance** - The distance the vehicle travels before before it is located by the system from the initial state
- **Recovery distance** - The distance the vehicle travels before it is located from a state where the system believes it is tracking the vehicle but is incorrect. This is tested by waiting until the system has located the vehicle and then artificially shifting the it to the other side of the map and measuring the distance it travels before the system locates it again. Because artificially shifting the vehicle will affect the other performance measures, recovery distance requires its own separate test.
- **Average Deviation Error** - The average estimate error while the system is deviating. A measure of the magnitude of the deviations.
- **Deviation Rate** - The average rate at which deviations occur measured as the percentage of the post-location system cycles which result in a deviation. A measure of how often deviations occur.

6.1.3 Default System and Testing Parameters

The section provides details about the default testing setup used in this chapter. Unless otherwise stated the test used the following system and testing parameter values.

System Parameters

- **Test Point Count** - The number of navigation data points that are compared to the map during a particle update. The default value is 30.
- **Test Point Spacing** - The distance between adjacent test points. The default value is 1m.

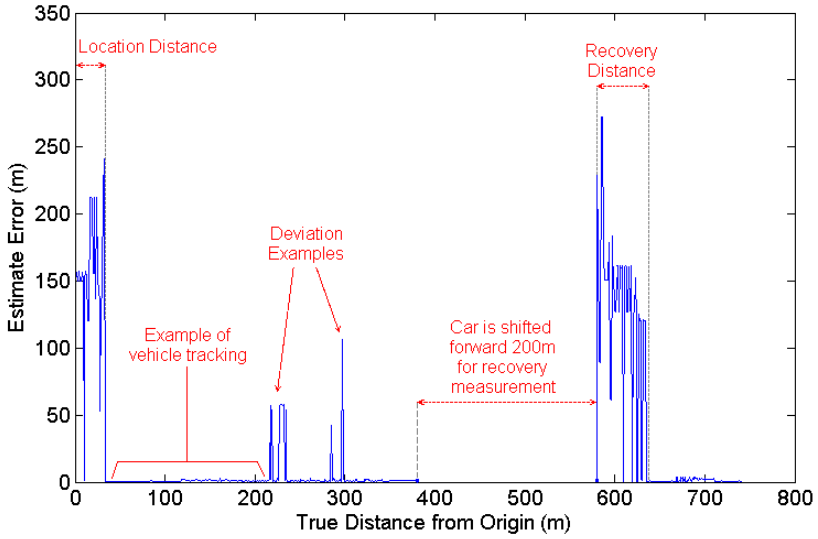


Figure 6.2 Visual representation of some of the testing terms and performance measures

- **Map Datasize** - The size of map data values in bits. The default value is 10 bits.
- **Map Data Spacing** - The Distance between adjacent map data values. The default value is 0.5m.
- **Particle Count** - The number of particles in the system. The default value is 50.
- **Alpha** - A system parameter used in resampling. See section 5.4.5 for details. The default value is 0.3.
- **Estimation Method** - The NULLS system uses one of two methods of location estimation: the max weight particle and the weighted average (see section section 5.4.4 for details). The default method is max weight particle

Testing Parameters

- **Location Thresholds** - Location is said to have occurred when the estimate error drops below 2m and stays below 2m for 10m of travel. These values are used in all tests.
- **Trials** - The number of times a test is performed before an average is taken. The

default is 100 trials.

- **Laps** - The number of times the vehicle travels around the location space in a 1-dimensional test. The default is 2 laps.

6.2 System Demonstration

This section demonstrates the NULLS system in action. Note that the results presented in this section are from single simulations and are not the average of multiple trials.

Figure 6.3 shows the estimate error, particle distribution and particle weights as the vehicle makes one circuit of a city block. As expected the estimate error is high at the beginning as the system has no navigation data to work with. After 100m the particles have converged on the correct location and track the vehicle well for most of the rest of the test with exceptions at 200m and 300m where deviations occur.

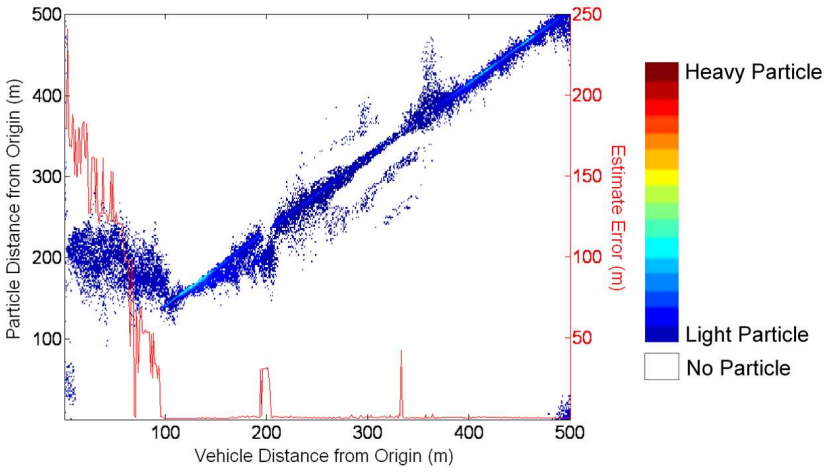


Figure 6.3 Paticle swarm and estimate error for vehicle location and tracking

Figure 6.4 shows the estimate error, particle distribution and particle weights as the system recovers from an incorrect track. This situation is simulated by running the system until it has a strong track on the vehicle and then artificially shifting it 150m forwards (this occurs at the point marked "Skip point" on Figure 6.4. The system is now in a state which closely resembles it having locked on to an incorrect position. After this point the particles quickly

diverge in an attempt to located the vehicle. After another 100m of travel the system has correctly locate the vehicle.

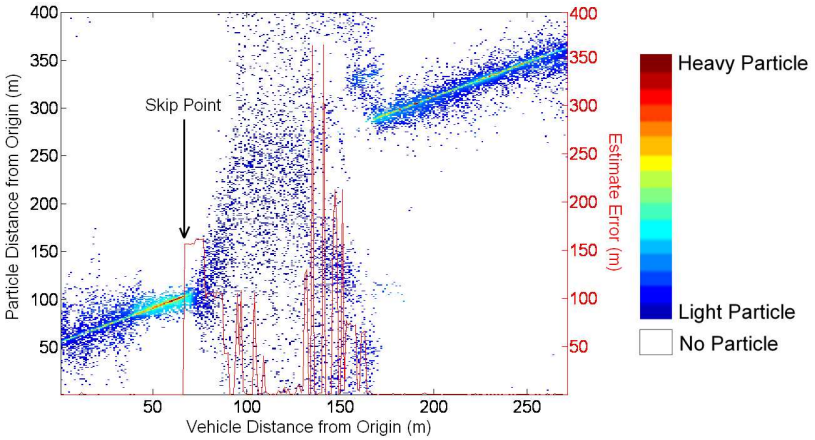


Figure 6.4 Particle swarm and estimate error for system recovering from an incorrect track

Figure 6.5 shows the estimate error signals of three simulations run on separate city blocks. This is provided to show that the NULLS system can function in a variety of place and has not been constructed to function in one particular area.

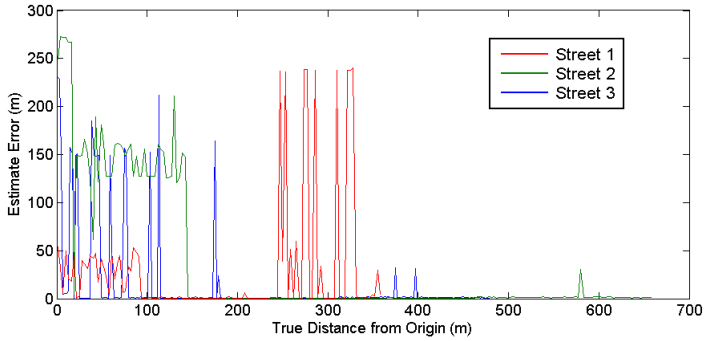


Figure 6.5 Particle swarm and estimate error for system recovering from an incorrect lock

6.3 System Parameters Analysis

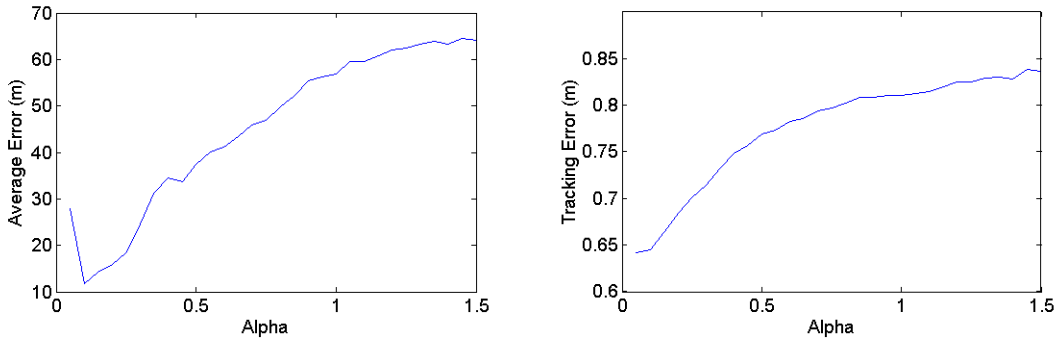
This section presents the results of a number of tests designed to show how each system parameter effects the system's performance. The location space used in the tests presented in this section was a single city block with a perimeter of 560m.

6.3.1 Algorithm Parameters

This section focuses on investigating the parameters that are part of the NULLS algorithms.

Alpha

Alpha is a system parameter used in the resampling step. In short, higher values of alpha lead to larger random movements of particles during a resample. Section 5.4.5 has details of exactly how alpha is used in the resampling algorithm. The effect of varying alpha on system performance was measured by running 30 simulations with alpha values ranging from 0.05 to 1.5 with a 0.05 step size. Figures 6.6 and 6.7 show the results of these tests.

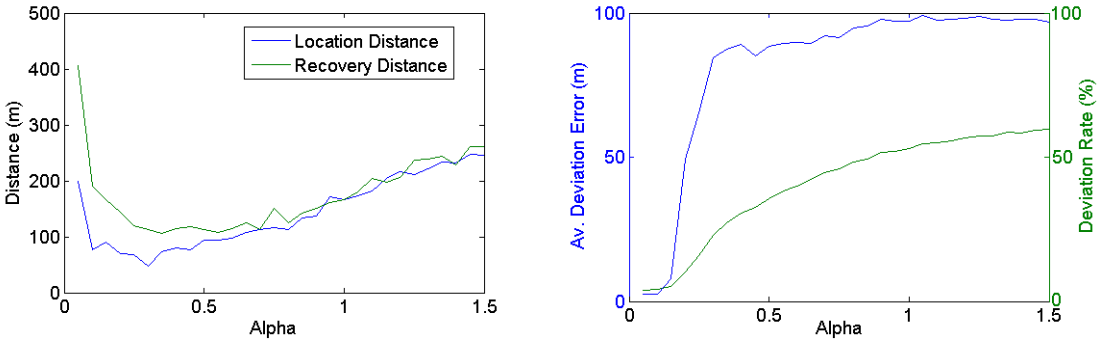


(a) Average error

(b) Tracking error

Figure 6.6 Average error and tracking error of alpha test

Low values of alpha, below approximately 0.25, lead to long location and recovery distances as shown in Figure 6.7(a) and indirecting in the average error of Figure 6.6(a). This is clearly because the particles are moving around very slowly and take longer to locate the vehicle. Above this approximate threshold all performance aspects degrade as alpha is increased because particles close to the vehicle with high weight values are moved too far during a resample to retain a strong track on the vehicle.



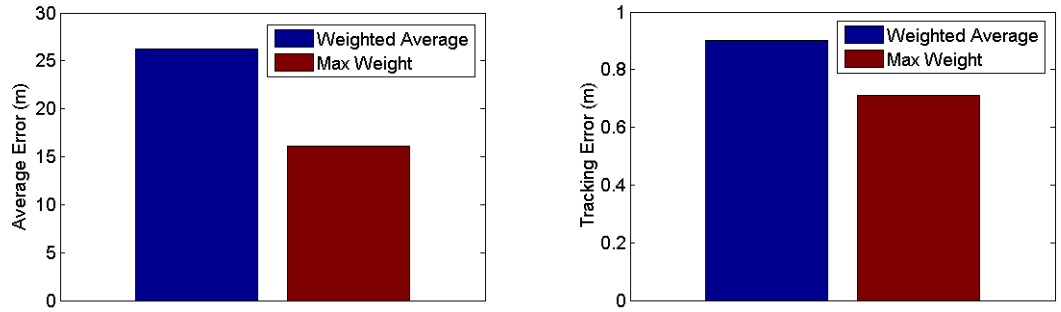
(a) Location and recovery distance

(b) Deviation rate and average error

Figure 6.7 Distance and deviation measures of alpha test

Estimation Method

The two estimations methods the NULLS system uses are the maximum weight particle and the weighted average. Details of how they work can be found in section 5.4.4. Figures 6.8, 6.9 and 6.10 show the results of a series of tests to see which method produces better location estimates.

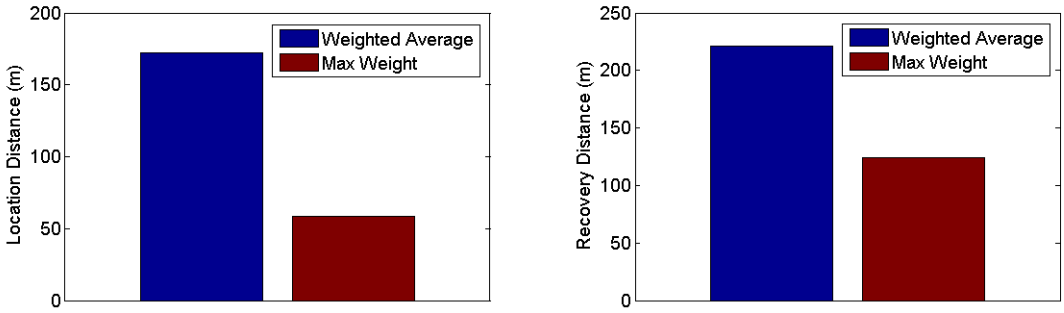


(a) Average error

(b) Tracking error

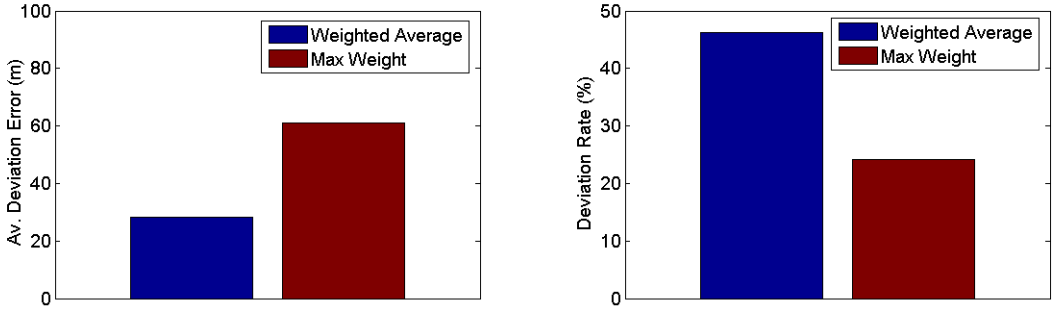
Figure 6.8 Average error and tracking error of estimation method test

The test shows that maximum weight particle method performs better than the weighted



(a) Location distance (b) Recovery distance

Figure 6.9 Location and recovery distance of estimation method test



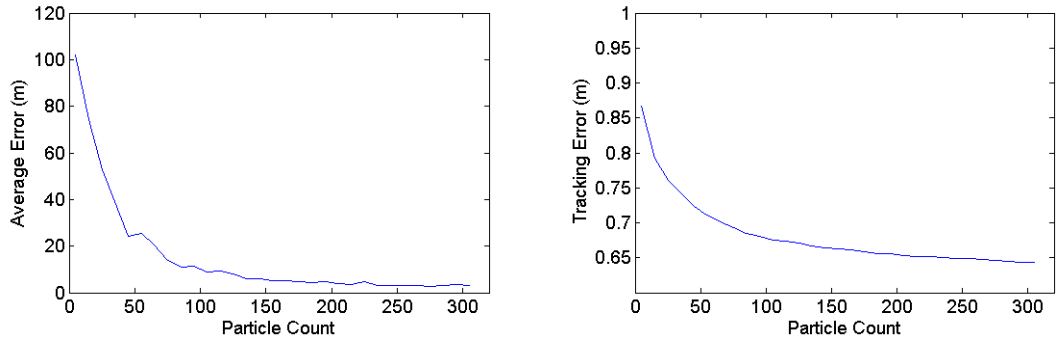
(a) Average deviation error (b) Deviation rate

Figure 6.10 Deviation measures of estimation method test

average method in all aspects but the average deviation error. This is not unexpected as when using the maximum weight particle method the estimated location can jump to any point in the location space due to a single particle being updated to have a misleadingly high weight an accidentally coincidence. The weighted average is far more resilient to this as its estimate is very likely to be close to the cluster of particles which generally forms around the vehicle's true location.

Particle Count

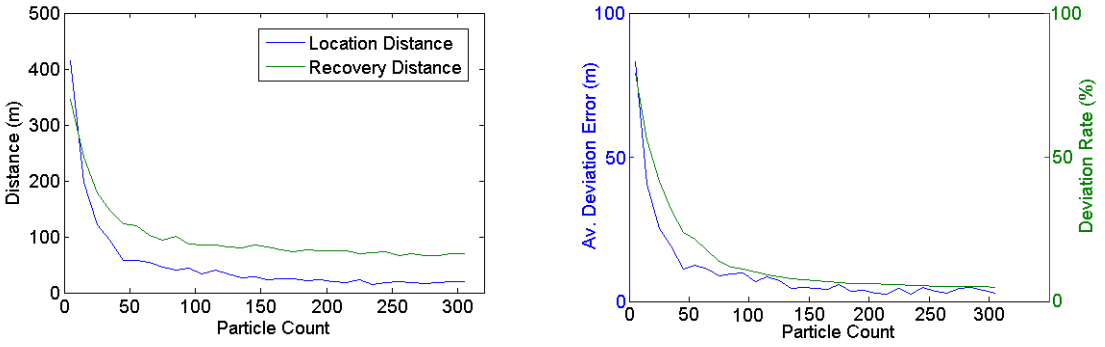
It is clear that all particle filters will see an improvement in performance as the number of particles is increased. Unfortunately this improvement comes at the cost of processing time as more particles means more particle updates, more particles to resample and more data to consider during the estimation step. The goal of this section's test is therefore not to see if system performance increases as particle count does but rather how it increases as a function of particle count and to determine a sensible particle count for the system setup used in this chapter. Tests were performed with a particle count ranging from 5 to 305 with a step size of 10 particles. The results are given in Figures 6.11 and 6.12



(a) Average error (b) Tracking error

Figure 6.11 Average error and tracking error of particle count test

Performance appears to increase only trivially after the particle count reaches 100. As specified at the beginning of section 6.3, the length of the location space used in this section is 560m. Dividing this approximation of a reasonable particle count (100) by the location space length (560m) gives a general particle rate of 17.86 particles per 100m of location space. Using a rough estimate of the total length of all the streets in Christchurch (5600km) this translates to a particle count of 1000160.



(a) Location and recovery distance

(b) Deviation rate and average error

Figure 6.12 Distance and deviation measures of particle count test

6.3.2 Data Parameters

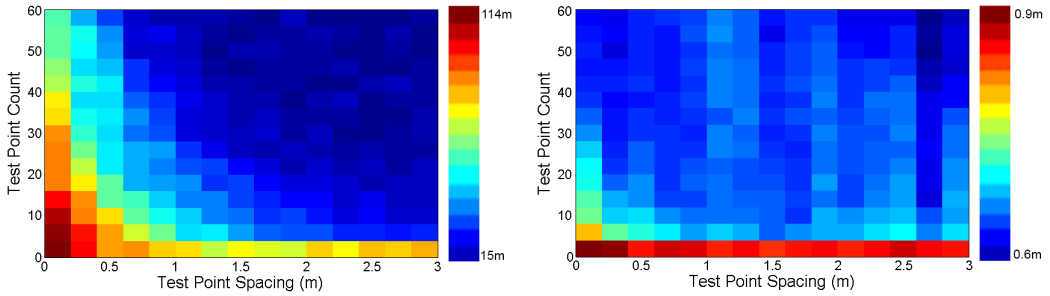
This section investigates the parameters related to the NULLS map and navigation data.

Test Points

The measurement step of the system cycle contain two parameters: the test point spacing, which is the distance between navigation data measurements, and test point count, which is the maximum number of test points the system uses in its calculations (older values are discarded when this maximum is reached (see section 5.4.2 for further details). Because these two parameters are so closely related they were tested together. The test point count parameter was varied between 1 and 57 with a step size of 4 (15 values in total). For each of these test point count values tests were performed using test point spacing values which ranged between 0.2m and 3m with a step size of 0.2m (15 values in total).

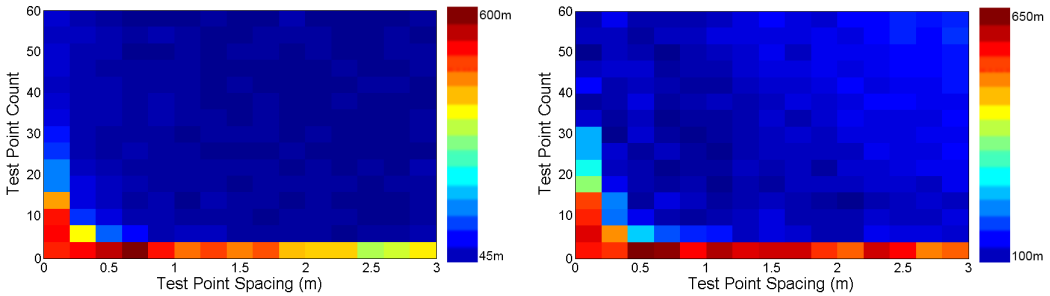
Figures 6.13, 6.14 and 6.15 show the results of this series of tests using Matlab's image function *IMAGESC* with its default "Jet" color map. Each figure contains a matrix of coloured squares. Each square represents a test. Their positions are defined by the values of the test point count and the test point spacing parameters which are marked on the x and y axes. Their colours show the value of the performance measure being presented in

the figure and are defined by the scale on the right.



(a) Average error (b) Tracking error

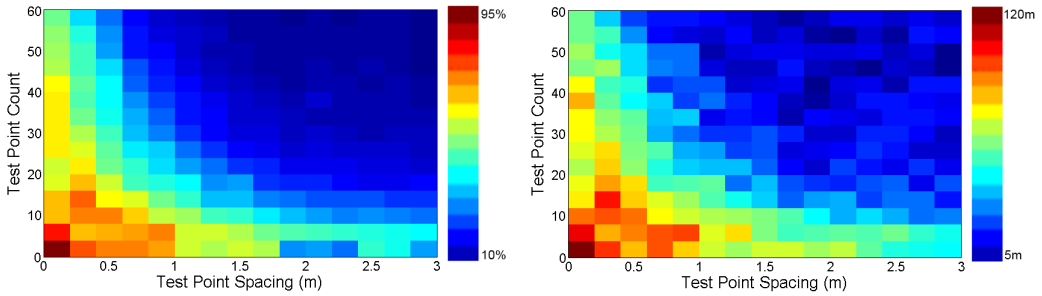
Figure 6.13 Average error and tracking error of test point test



(a) (b)

Figure 6.14 Location and recovery distances of test point test

The general trend shown by these tests is that overall performance is improved as test point count and test point spacing increase, or in more general terms, as more data is used over a wider area. What is most surprising is how little the tracking error, location distance and recovery distance measures change after relatively low parameter thresholds (of approximately 0.5m for the test point spacing and 10 for the test point count) are reached. It is important to note that the map used in this test has a data spacing of 0.5m, as defined in section 6.1.3, which may account for the 0.5m performance threshold on the test point spacing.



(a) Deviation Rate

(b) Average Deviation Error

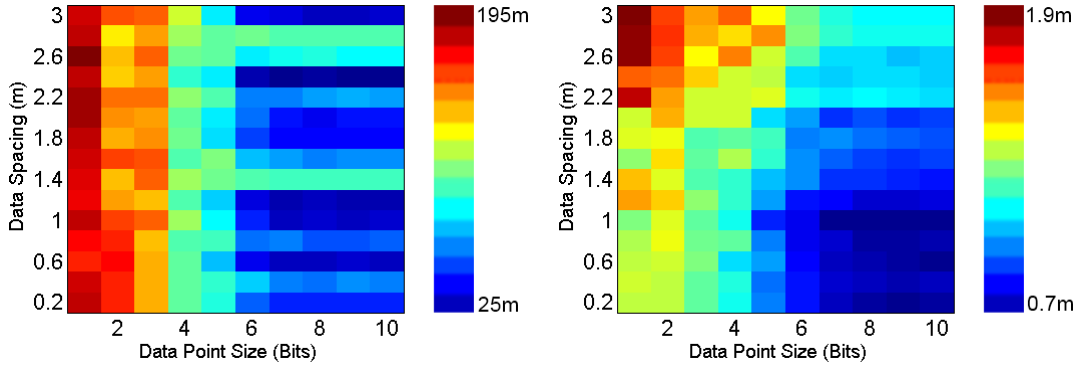
Figure 6.15 Deviation measures of test point test

Map Data Rate

The two main parameters of the map generation process are the data spacing and size of each data point in bits. See chapter 3 for details. Because these two parameters are so closely related they were tested together. The data spacing parameter was varied between 0.2m and 3m with a step size of 0.2m (15 values in total). For each of these data spacing values tests were performed using different data point size values ranging between 1 bit and 10 bits.

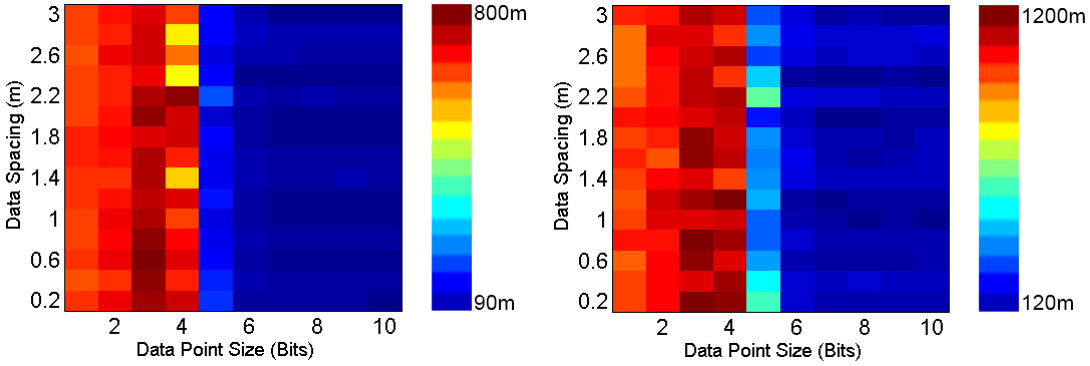
As in section 6.3.2 Figures 6.16, 6.17 and 6.18 show the results of this series of tests using Matlab's image function *IMAGESC* with its default "Jet" color map. Each figure contains a matrix of coloured squares. Each square represents a test. Their positions are defined by the values of the data spacing and data point size parameters which are marked on the x and y axes. Their colours show the value of the performance measure being presented in the figure and are defined by the scale on the right.

While these results contain significantly more noise than those of the other parameter tests some trends are still clear. Tracking error decreases both as data point size increases and data spacing deceases. This can be interpreted as an increase in accuracy as the data rate increases, which makes perfect sense. Data spacing does not appear to have any effect on



(a) Average error (b) Tracking error

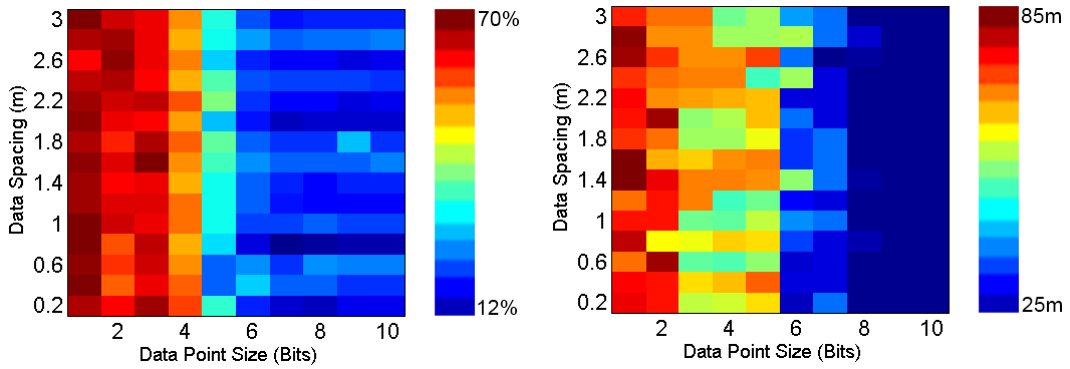
Figure 6.16 Average error and tracking error of map data rate test



(a) (b)

Figure 6.17 Location and recovery distances of map data rate test

the other performance measures. This is most likely because they are generally based on whether the estimate error is below the 2m ”location threshold” defined in section 6.1.3 and do not consider more accurate estimates. An exception to this is the average error measure which is probably dominated by the large pre-location estimate errors and the deviations so that it does not show the increased accuracy that lower data spacing provides. The data point size appears to have a threshold, of approximately 6 bits (varying slightly between performance measures), at which point performance is greatly increased



(a) Deviation Rate

(b) Average Deviation Error

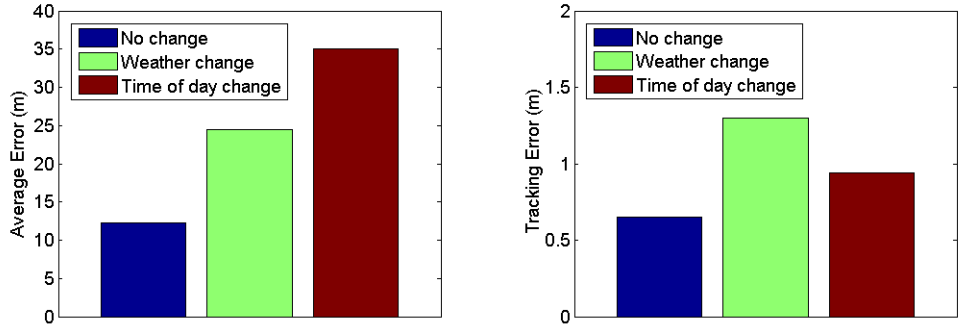
Figure 6.18 Deviation measures of map data rate test

but does not appear to change significantly beyond that point. Using a rough estimate of the total length of all the streets in Christchurch (5570km), a data point size of 6 bits and a data spacing of 2m an uncompressed luminance map of Christchurch would required 1.992mb of memory.

6.4 World Variations

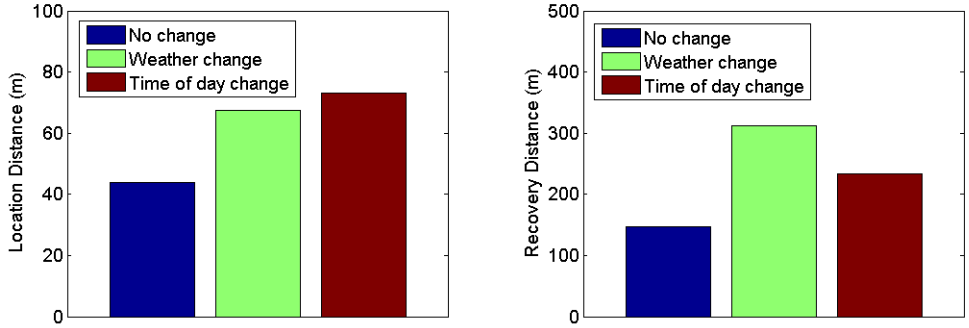
This section presents the results of a series of tests which measure the negative impact of varying weather and time of day conditions on the system's performance. Four datasets were collected from the same city block. The first dataset was collected at 10 a.m. on a clear January morning and was used as the map throughout the tests. The second dataset was collected immediately after the first so that there was little or no change in the weather conditions and the position of the sun. This set was used as the control navigation set, to which the others were be compared. The third set was recorded at 6 p.m. on the same cloud-free day and was used as a navigation dataset to see how a change in the sun's position would affect the system's performance. The final dataset was recorded at 10 a.m. on an overcast day and was used as a navigation dataset to see how a significant change in weather conditions would affect system performance. Three performance tests were then

carried out each using the same map and one of the three different navigation datasets. The results of these tests are presented in Figures 6.19, 6.20 and 6.21.



(a) Average error (b) Tracking error

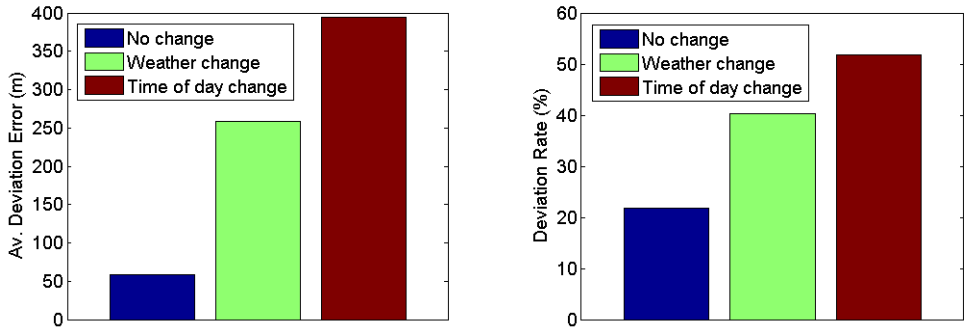
Figure 6.19 Average error and tracking error of world variations test



(a) Location distance (b) Recovery distance

Figure 6.20 Location and recovery distance of world variations test

As might be expected, performance was significantly lower in the weather and time of day variation tests than in the control test. While the weather variation test generally had better results than the time of day variation test the difference is not large enough to say which factor is likely to be a bigger challenge for the NULLS system in general based on a test of this scale. Such a decision would require many more datasets from a variety of



(a) Average deviation error

(b) Deviation rate

Figure 6.21 Deviation measures of world variations test

different times of day, weather conditions and location spaces.

6.5 2-Dimensional Location Demonstration

This section demonstrates the ability of the 2-dimensional location system to correctly determine which path the vehicle took after passing through a fork in the location space. Figure 6.22 shows a very simple 2-dimensional location space with three street segments and one fork (marked in light blue).

Figure 6.23 shows four snapshots of particles and their weights on three different street segments during a simulation using the location space in Figure 6.22 where the vehicle moves from Street 1 to Street 2. The snapshots were recorded four cycles apart. The purpose is to demonstrate the movement of particles during a branching operation. Figure 6.23(a) was recorded just before the vehicle leaves Street 1 and turns onto Street 2. The particles are clustered around the end of Street 1 with only one stray particle on each of the other two streets. In Figure 6.23(b) the vehicle has turned onto Street 2 and the particles have shifted from Street 1 to both Street 2 and Street 3 in approximately even quantities **unclear**. Figures 6.23(c) and 6.23(d) shows the particles on Street 3 reducing in quantity

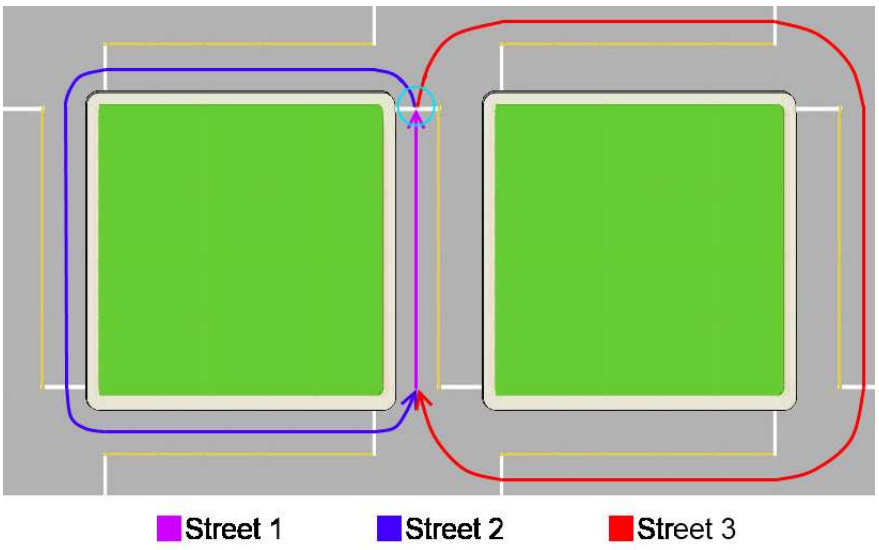
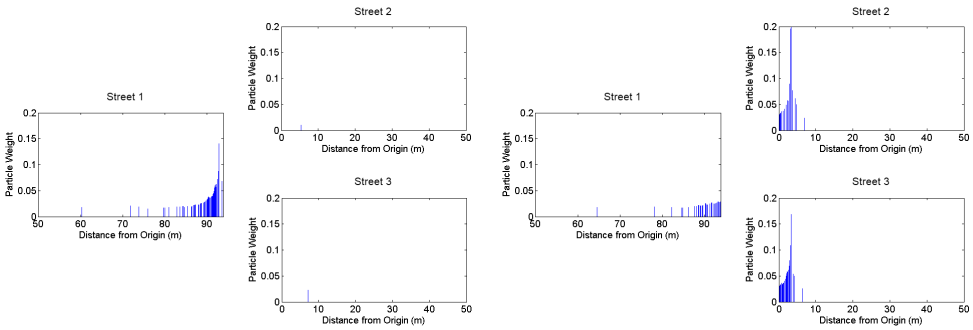


Figure 6.22 Illustration of the location space used in the 2D location demonstration

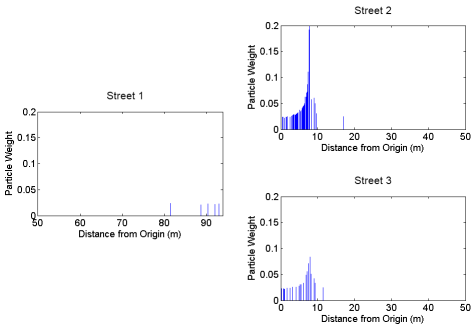
to only four as the newly gathered data show the system that vehicle turned onto street 2 and not Street 3.

While the testing performed for the 2-dimensional location system was much less extensive than that performed for 1-dimensional location, it nevertheless demonstrates that the system can correctly identify which of a few candidate streets has been entered. It is likely that the 2-dimensional location system will be sensitive to at least some of the system parameters but more extensive testing is required.

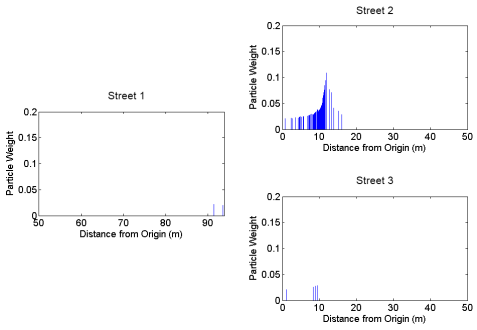


(a) t = n

(b) t = n + 4



(c) t = n + 8



(d) t = n + 12

Figure 6.23 Particle weights on 3 street segments at 4 points during 2D location simulation (t refers to the current system cycle)

Chapter 7

Conclusions and Future Work

7.1 Conclusions

A system using luminance data and particle filtering to located and track vehicles in an urban environment was developed and tested. During the course of this project many of the intended goals were achieved. A luminance data recorder was developed, and it proved acceptable for providing the offline analysis software with real world data with which to test the concept. Luminance map generation was achieved and a powerful location algorithm was developed based on a particle filter. Detailed offline testing of the system was carried out. While the tests used very small location spaces when compared to realistic scenarios the ability of the system to track a vehicle from a known location, appears to be both strong, and independent of the size of the location space. Locating vehicles from a completely unknown initial position was shown to be possible in the small test location spaces but it is clear that the computational cost of this task will increase rapidly, as the size of the location space increases. The memory requirements of the luminance maps appears to be encouragingly low with a rough estimate of a map of Christchurch needing less than 2MB of memory.

7.2 Future Work

This section suggests possible areas for future research into the NULLS system.

7.2.1 Data Recording

Recording luminance map data for a large location space would be a very expensive and time consuming process. This sections suggests two methods for reducing the effort involved.

GPS Equipped Vehicle Fleet

Companies that use large fleets of vehicles, such as couriers and taxis, generally have their fleets equipped with GPS receivers. NULLS data recorders could be installed on these vehicles and use their GPS to track their positions while recording luminance data. This would allow luminance maps to be gradually built up, without the company developing NULLS, spending the vast amount of resources required to drive a vehicle down every street in the desired location space.

Google Street View

Google Street View provides photographic data from the sides of every street in New Zealand and many other countries. The data was recorded by vehicles travelling on the roads with cameras oriented perpendicularly to the to the direction of motion, just as the NULLS luminance sensor is. This makes Google Street View an ideal source of NULLS map data. It would be a relatively simple signal processing task to convert the photographic data into a 1-dimensional signal matching the NULLS map data format. This would vastly reduce the effort required to build NULLS maps.

7.2.2 System Improvements

Environmental Data

In addition to the position and luminance data, collected and analysed in this project, the time of day and the ambient light conditions could also be measured during both the map data collection and the location estimation processes. Time of day could be measured from the vehicle's clock and ambient light levels by using a wide-angle light sensor, ideally facing upwards. This additional data could be used to counteract the negative impact of changing environmental conditions. An intelligent learning system, such as a neural network, could be employed to determine how to best make use of this data. This would involve finding a function to compare the environmental conditions of the map and navigations datasets and suggesting how the navigation luminance data should be adjusted to account for the differences.

Location Estimation Methods

As discussed in section 6.3.1 both the maximum weight particle and the weighted average location estimation methods have downsides. An improved estimation method could not only incorporate both methods, in an attempt to reduce their respective problems, but could also consider position estimates from previous system cycles.

Parameter Optimization

Section 6.3 showed how each system parameter can have a significant effect on system performance. Because of the dependence the parameters are likely to have on one another finding optimal values for all of them will be no simple task. A genetic algorithm or a similar intelligent optimisation system would likely prove invaluable to this process.

7.2.3 Hybrid Navigation Systems

While it has been shown that the NULLS system can be used to locate vehicles in a small location space the computational requirements will increase steadily as the size of the loca-

tion space increases. The estimation presented in section 6.3.1 for the numbers of particles required to retain similar performance levels for location within the city of Christchurch suggests that the computational power requirement will be unrealistically high in such a location space. While it is very possible that lower system performance would be acceptable, it appears unlikely that NULLS is suitable as the only location method used by a vehicle navigation system. The ability for NULLS to track a vehicle once its approximate position is known, is however, very good and has a relatively low computational cost. Therefore the use of another location method in conjunction with NULLS could prove to be commercially viable.

As mentioned in chapter 1 inertial navigation systems (INS) are generally very effective at tracking vehicles but suffer from integration drift - where the estimate error slowly increases throughout the journey. This makes INS ideal candidates for using NULLS as an auxiliary navigation system. The INS would provide accurate motion data in the short term, thereby allowing the NULLS system to concentrate on a small, shifting location space, and update the INS with absolute location data thus eliminating the drift errors.

References

[BF99] Matthew Barth and Jay Farrell. *The global positioning system and inertial navigation*. McGraw-Hill, New York, 1999.

[Dan99] Peter H. Dana. "Global positioning system overview," [Online] Available: `http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html` [Accessed: Feb. 2011], 1999.

[DdFG01] A. Doucet, N. de Freitas, and N. Gordon. "Sequential monte carlo in practice," *Springer-Verlag*, 2001.

[Dev03] Analog Devices. "Linear output magnetic field sensor," *AD22151 datasheet*, 2003.

[DKZ+03] P. Djuric, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo, and J. Miguez. "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, September 2003.

[GGB+02] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. Nordlund. "Particle filters for positioning, navigation and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, February 2002.

[GWA07] Mohinder S. Grewal, Lawrence R. Weill, and Angus P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. John Wiley & Sons, Hoboken, N.J, 2nd edition, 2007.

[Kel94] Alonzo Kelly. *Modern inertial and satellite navigation systems*. Robotics Institute, Carnegie Mellon University, Pittsburgh, Pa, 1994.

[KFM04] C. Kwok, D. Fox, and M. Meila. "Real-time particle filters," *Proceedings of the IEEE - Special Issue on Sequential State Estimation*, vol. 92, no. 2, 2004.

[Lab06] Charmed Labs. "A powerful robotics solution for university and high school educational and hobbyist markets," `http://www.charmedlabs.com/index.php?option=com_content&task=view&id=29`, 2006.

[MAX99] MAXIM. "3.0v to 5.5v, low-power, up to 1mbps, true rs-232 transceivers using four 0.1f external capacitors," *MAX3232 datasheet*, 1999.

[Oli06] Olimex. "Development board for arm at91sam7s64," `http://www.olimex.com/dev/index.html`, 2006.

[Sol07] Texas Advanced Optoelectronic Solutions. "High-sensitivity light-to-voltage converter," *TSL257 datasheet*, September 2007.